Copyright © 2014 TenAsys Corporation,
All Rights Reserved

eVM® for Windows 2 User Manual

Copyright © 2014 TenAsys Corporation All Rights Reserved

No part of this manual may be copied, duplicated, reprinted, and stored in a retrieval system by any means, mechanical or electronic, without the written permission of the copyright owner.

First Printing: January 2014

Published in the United States by

TenAsys Corporation

1400 NW Compton Drive

Suite 301

Beaverton, OR 97006, USA

Attempts have been made to properly reference all copyrighted, registered, and trademarked material. All copyrighted, registered, and trademarked material remains the property of the respective owners.

The publisher, author, and reviewers make no warranty for the correctness or for the use of this information, and assume no liability for direct or indirect damages of any kind arising from the information contained herewith, technical interpretation or technical explanations, for typographical or printing errors, or for any subsequent changes in this article.

The publisher and author reserve the right to make changes in this publication without notice and without incurring any liability.

Windows® and Windows Vista® are registered trademarks of Microsoft® Corporation.

TenAsys®, INtime®, and iRMX® are registered trademarks of TenAsys® Corporation.

eVM® is a registered trademark of TenAsys® Corporation.

Intel® is a registered trademark of Intel Corporation.

All other company names and products herein may be trademarks of their respective owners.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

Table of Contents

1 Introduction 8
1.1 Overview of eVM for Windows 8
2 Quick Start Guide 11
3 Product Technical Description 12
3.1 What is Real-Time embedded virtualization? 12
3.2 Theory of Operation of eVM 12
3.2.1 What is eVM 12
3.2.2 Introduction to virtualization 12
3.2.3 eVM and Virtualization technology 13
3.2.4 eVM system requirements (brief overview) 13
3.2.5 Performance: How eVM addresses RTOS / Windows Application Needs 14
3.2.6 Communications between Windows and the Guest: Virtual Device Interfaces 14
3.2.7 Communications between Windows and the Guest: Shared Memory 15
3.2.8 eVM Architecture: Partitioning the PC between Windows and Guest OS 15
3.2.9 Resources that are not Isolated between Windows and the Guest OS 17
3.2.10 Device Management 17
3.2.11 What Happens when a Device is assigned to eVM 22
4 Installation 23
4.1 Installation of eVM for Windows 23
4.2 Virtual Ports 41
4.2.1 Installation of Virtual Serial Ports 42
4.2.2 Configuring the Virtual Ethernet Device 47
4.3 Running the Sample Guest RTOS, iRMX 55
4.4 Installation of a Guest OS 57
4.4.1 Configure your existing system 57
4.4.2 Make a disk image of your system 58
4.4.3 Make a guest configuration 58
4.4.4 Installing your operating system from installation materials 59
4.5 Specific steps for certain Guest OSes 59
4.5.1 Installation and configuration of QNX 6.4 59
4.5.2 Installation and configuration of Windows CE 62

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

4.6 Driver configuration and setup 63
4.6.1 eVM Device Manager 63
4.6.2 Passing a Legacy IRQ Device to eVM 68
4.6.3 Conflict Example when Passing Legacy IRQ Device to eVM 77
4.6.4 Passing an MSI Device to eVM 79
4.6.5 Bus Master Warning Example 82
4.6.6 Passing a Device back to Windows 84
4.7 Shared Memory Interface 88
4.7.1 Description 88
4.7.2 Configuration 88
4.7.3 Shared Memory Usage 90
4.7.4 PCI configuration space definition 91
4.7.5 Register definition 91
4.7.6 Typical operation 92
A. Production installation 94
A.1 Saving the configuration with the /e: option 94
A.2 Restoring or importing a known configuration 95
A.3 Assigning devices to eVM on the command line 97
PCI devices 97
A.4 Command Line Installation of Virtual Serial Ports 97
A.5 Debug logging 97
B. eVM Virtual Machine 99
B.1 System Requirements 99
B.2 Base Virtual Machine Specification 99
CPU 99
Memory 99
Interrupt controller (8259A PIC) 100
System Timer (8254 PIT) 100
CMOS & Real Time Clock 100
Port 92 100
Serial ports (16550 UART) 101
DMA controller (8237 DMA controller) 101

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

Floppy Disk Controller 101
IDE controller 101
PCI Host Bridge (440FX Host Bridge, PCI device 0) 101
ISA Bridge (440FX ISA Bridge, PCI ID 8086/7000) 101
EIDE Controller (PCI ID 8086/7010) 101
ACPI Controller (PIIX4 ACPI Controller, PCI ID 8086/7113) 102
B.3 Optional Virtual Machine Components 102
Ethernet controller 102
Shared memory device 102
B.4 BIOS 102
C. Guest OSes Currently Enabled: 103
D. VtProbe 104
E. Modifying device drivers for bus-master devices 106
E.1 Discovering the guest physical base address 106
E.2 Using the guest physical base offset value 106
F. eVM API calls 107
F.1 Include the API in your application 107
F.2 Types 107
F.3 API Calls 107
General calls 107
Command and Control calls 107
VMM Management calls 108
Configuration Management calls 108
Shared Memory Management 109

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

List of Figures

Figure 1.1 eVM for Windows 9
Figure 3.1 Memory map for a 32 bit installation 15
Figure 3.2 Memory map for a 64 bit installation 16
Figure 3.3 Interrupt management 19
Figure 4.1 VT-x not supported message 24
Figure 4.2 Installation Wizard 25
Figure 4.4 Release Notes ..... **Error! Bookmark not defined.**
Figure 4.3 License Agreement 27
Figure 4.4 Release Notes 28
Figure 4.5 - Select eVM Installation Folder 29
Figure 4.6 - Choosing Standard Installation Type 30
Figure 4.7 - eVM Activation 31
Figure 4.8 – eVM Installation Verification 32
Figure 4.9 – eVM Installation Wizard Progress Dialog 33
Figure 4.10 - eVM Installation Wizard Successful Installation 34
Figure 4.11 TenAsys Driver loading notification 34
Figure 4.12 - eVM Installation Log 35
Figure 4.13 - Restart Prompt 35
Figure 4.14 - eVM System Tray Icon 36
Figure 4.15 Launching the eVM Control Panel from the system tray 36
Figure 4.16 eVM Configuration Panel 37
Figure 4.17 - eVM Control Panel with iRMX Guest RTOS Installed 38
Figure 4.18 Device Manager showing the Windows devices that can be passed to eVM 39
Figure 4.19 Device Manager showing all the devices, including ones that cannot be passed to eVM (the guest OS) 40
Figure 4.20 Windows Device Manager 41
Figure 4.21 eVM Control Panel Showing System Configuration with No Virtual Serial Ports 42
Figure 4.22 - eVM Control Panel, Virtual Serial Port Added 43
Figure 4.23 Virtual Serial Port COM Port Selections 44
Figure 4.24 Saving eVM Control Panel Configuration 45
Figure 4.25 - Windows Device Manager Showing Virtual Serial Port 46
Figure 4.26 - eVM Control Panel Showing Guest Serial Device Settings 47
Figure 4.27 - Windows Control Panel, Select Network 48
Figure 4.28 - Windows Network and Sharing Center 49
Figure 4.29 - Windows Network Connections 50
Figure 4.30 - TenAsys Virtual Ethernet Adapter, Windows Local Area Connection Properties 51
Figure 4.31 - Selecting TenAsys Virtual Ethernet Adapter TCP/IPv4 Properties 52
Figure 4.32 - Setting the Static IP Address for the TenAsys Virtual Ethernet Adapter 53
Figure 4.33 - eVM Control Panel, Guest OS Virtual Ethernet Device Configuration 54

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

Figure 4.34 - eVM Control Panel, Selecting the Guest OS 55
Figure 4.35 - Guest RTOS, iRMX, Booting in the Virtual Machine 56
Figure 4.36 - eVM Control Panel, Stopping the Guest OS 57
Figure 4.37 – Launching the *eVM Device Manager* from the *eVM Configuration Panel* 64
Figure 4.38 - eVM Device Manager launch from Widows Vista or Window 7 65
Figure 4.39 - eVM Device Manager launch from Windows XP 65
Figure 4.40 - eVM Device Manager showing all the devices that can be passed. 66
Figure 4.41 - eVM Device Manager – how to change to show all the devices. 67
Figure 4.42 - eVM Device Manager Showing All Devices 68
Figure 4.43 - COM1 properties in eVM Device Manager 69
Figure 4.44 - Pass Device Using Pass to eVM Button 70
Figure 4.45 - Pass Device Using Action Dropdown 71
Figure 4.46 - Device Pass Request Selected 72
Figure 4.47 - Using the Save Configuration Button 73
Figure 4.48 - Using Dropdown to Save the Configuration 74
Figure 4.49 Warning that you need VT-d support if the device is a bus master and you want to DMA to it. 75
Figure 4.50 - Platform Restart Required to Apply New Device Configuration 76
Figure 4.51 - Device Configuration Change Applied 77
Figure 4.52 - eVM Device Manager Showing Windows USB Controllers 78
Figure 4.53 - eVM Device Manager Flagging a Conflict 79
Figure 4.54 - Right-Click a Device to Pass it to eVM 80
Figure 4.55 - MSI Capable Device Pass Request 81
Figure 4.56 - MSI Network Adapter Device Passed to eVM 82
Figure 4.57 - VT-x Support Enabled but VT-d Support Disabled 83
Figure 4.58 - Bus Master Warning for no VT-d Support 84
Figure 4.59 - Select the Network Adapter and Pass It Back to Windows 85
Figure 4.60 - Request to Pass the Network Controller back to Windows 86
Figure 4.61 - Restart Request to Apply Device Configuration Changes 87
Figure 4.62 - Network Adapter Back Under Windows Control 88
Figure 4.63 Setting size of *Shared Memory* 89
Figure 4.64 Setting Shared Memory IRQ for guest application. 90
Figure 4.65 - eVM Configuration for My Old OS 95
Figure 4.66 - eVM Configuration 96
Table B 1 Virtual IRQ's 100
Figure D. 1 - VtProbe, Hardware Platform Doesn't Support eVM 104
Figure D. 2 - VtProbe: Hardware Supports Virtualization Technology 105

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

1 Introduction

1.1 Overview of eVM for Windows

eVM for Windows provides a virtual machine environment hosting an embedded or real-time operating system alongside Microsoft Windows. eVM for Windows runs in parallel to Windows on any Intel multi-core or multi-threaded PC platform that features Intel Virtualization Technology (VT). Both Windows and the embedded OS run natively, as if they were running on their own dedicated platforms. The result is that, moving an embedded OS plus the application running on it to an eVM platform is a simple process that provides instant access to modern multi-core computing platforms, as well as, access to all of the applications, the development environment, and the rich graphic user interface capabilities that are available under Windows. The key features of eVM for Windows are:

- Memory partitioning such that the guest embedded OS and Windows maintain their integrity – there is no chance for the one operating system to corrupt the other's memory space.
- Memory re-mapping allows both Windows and the guest embedded OS to run native code without any modification to drivers and other memory-mapped utilities.
- Hardware access and interrupt mapping of specific I/O's to the embedded OS, so that they can continue to be served with legacy drivers under the embedded OS.
- Means of emulating legacy I/O's to minimize the rewriting of legacy code. Thus a legacy ISA-based system can easily be migrated to a state of the art and less expensive single-board computer with the simple re-direction of the ISA peripheral to an equivalent on-board PCI device.
- Access to Windows I/O resources by the embedded OS via virtual I/O device drivers.
- Shared Memory block to enable Windows and Guest OS to exchange information – supported by API.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

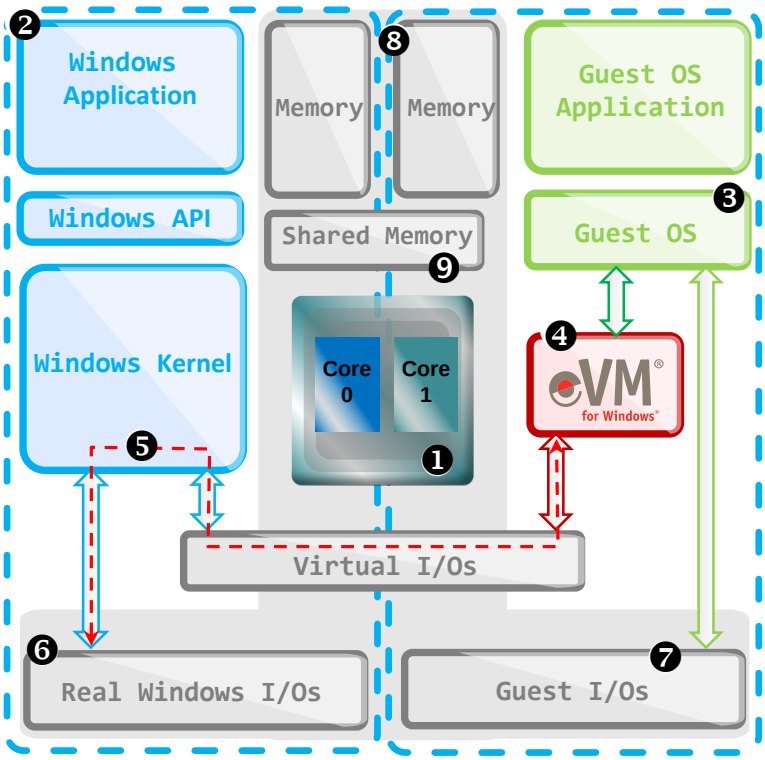


Figure 1.1 eVM for Windows

1. Multi-core Intel Processor platform supporting Virtual technology
2. Un-modified Windows 8, 7, Vista or XP (32- oir 64-bit versions) – running on a dedicated processor Core.
3. Un-modified guest OS – Real-time or other PC compatible embedded OS – running on dedicated processor Core,
4. eVM for Windows runs on its own independent of Windows and the guest OS.
5. Virtual I/O device interface allows guest OS to use Windows supported I/O devices.
6. I/O devices that are under the control of Windows ***and*** Memory that is partitioned for the exclusive access by Windows.
7. I/O devices that are under the control of Windows
8. Memory that is partitioned for the exclusive access by the Guest OS and Windows.
9. Shared memory with supporting API to enable Guest OS and Windows applications to pass messages to each other.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

2 Quick Start Guide

It is important to understand the theory behind eVM virtualization before attempting to install and use this product. The next section provides the technical details and background of virtualization and eVM. If you already understand the technical specifics of eVM or just can't wait to get your hands on it, sections 4.1 through 4.4 are quick start guide. Section 4.1 will walk you through the eVM installation process. Section 4.2 will describe installing virtual serial ports. Section 4.2.2 will describe configuring the virtual Ethernet device. Finally, Section 4.3 will guide you through executing the sample guest OS, iRMX, that is provided with the installation.

Important: ***If you haven't verified that your target platform has the appropriate VT-x and VT-d features to run eVM for Windows, be aware that the installation runs a verification tool called VtProbe; and the installation will be aborted, if the target platform does meet the virtualization criteria. The stand-alone VtProbe tool (VtProbe.exe) can be downloaded from*** www.tenasys.com***. VtProbe can be used to verify your platform's compatibility with eVM before you begin the product installation.***

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

3 Product Technical Description

3.1 What is Real-Time embedded virtualization?

Embedded virtualization is a methodology used to partition the hardware resources of a platform between a host operating system and a guest operating system so critical hardware resources can be dedicated each operating system. Embedded virtualization is optimized to minimize the virtualization overhead for the guest OSes. Real-time embedded virtualization ensures that critical hardware resources are expressly controlled by the guest OS with minimum, deterministic device latency and minimal CPU overhead.

3.2 Theory of Operation of eVM

The eVM virtualization platform provides a virtual machine that hosts real-time and embedded operating systems running alongside Microsoft Windows. The virtual machine manages the hardware resources to provide a guest OS environment whereby a real-time guest OS can still run as a bounded, real time operating system without modification.

3.2.1 What is eVM

eVM is an embedded virtualization environment that is hosted on the Windows platform to allow consolidation of the real-time control capability of a guest OS with the Windows capability on a common hardware platform.

eVM uses the Windows platform to create a virtual machine capable of supporting a real-time guest OS that can run as a guest OS without degradation in real time performance. eVM is also designed to have minimal impact on the Windows platform, as well. Some embedded products split processing between two computers: one for real-time and the other for Windows interface. Using the eVM virtualization environment allows you to take advantage of both the Windows services and the real-time guest OS services on the same hardware platform. This provides a reduction in hardware cost by not having separate hardware systems, one to support Windows and the other to support the real-time OS. Windows and the guest OS are also tightly connected in the eVM environment, which provides some advantages for communications and interaction between Windows and the real-time guest OS.

3.2.2 Introduction to virtualization

Virtualization is the abstraction of computing resources. In this specific case this abstraction separates the target operating system from its physical resources. When supporting multiple operating systems on the same platform, some interfaces for a given operating system are emulated, and some are mapped directly to a given hardware interface. The emulation of the virtual interfaces is the key to successful virtualization.

Many of Intel's new processor and chipset families have been specifically designed with hardware capabilities that provide virtualization capability and improve the efficiency of the virtual environment through hardware support. These hardware features are known as Intel

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

Virtualization Technology. Two of the pieces of this Virtualization Technology that are used by eVM are VT-x, Intel Virtualization Technology for processor support, and VT-d, Intel Virtualization Technology for Directed I/O, which is chipset technology.

3.2.3 eVM and Virtualization technology

eVM is designed to take advantage of the Intel Virtualization Technology to run a real-time guest OS without having to modify the RTOS, drivers and interfaces. All that is required is to configure the guest OS for the virtual machine. All important interfaces are handled directly by the guest OS and the applications running on the guest OS, and not through a virtualized layer. This allows the guest OS to run in real-time without modification or customization to the virtual environment.

eVM partitions the platform. Some resources are driven completely by Windows, and some resources are driven by the guest OS. To maximize performance, eVM dedicates a whole CPU hardware thread to the guest OS environment. The hardware thread is not virtualized, in that the dedicated hardware thread is available to run the guest software whenever the guest is ready to run. Devices that are critical to the applications running on the guest OS are dedicated to the guest OS and are directly accessible to the applications running on the guest OS without going through a virtualization layer. The VT-x technology assists the CPU to translate memory addresses in accessing the memory dedicated to the guest OS.

There are also two virtual interfaces for communication between Windows and the quest OS. There is a virtual serial interface and a virtual Ethernet interface that emulate wired links between the operating systems. A single virtual Ethernet link and up to four virtual serial links are supported.

Finally, there are devices on the virtual machine that are completely emulated, because Windows has control of the real PC hardware. Devices such as CMOS, access to the PCI bus, timers, and basic core PC functionality fall into this category.

3.2.4 eVM system requirements (brief overview)

The following lists the hardware requirements for a host platform that can run eVM:

- A multicore Intel processor or a single core processor with hyper-threading.
- Intel Virtualization Technology for x86 CPUs, (Intel VT-x), and a BIOS that supports enabling VT-x.
- The default amount of RAM that will be taken away from Windows and dedicated to the guest OS and eVM is 320 MB. The minimum memory requirement would be 64 MB (for eVM) plus the minimum memory requirement for the guest OS.
- In the case where the guest OS will access a device which uses DMA, it is desirable to have Intel Virtualization Technology for Directed I-O,(Intel VT-d), which eVM will use for DMA remapping. If VT-d is supported by the chipset, then the platform must also have a BIOS that enables VT-d.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

It can be difficult to determine which PC platforms and motherboards have VT-x and/or VT-d support. The specific details about the Intel Virtualization Technology support and the BIOS support is not well documented on much of the hardware available, today. It may take some discussions with technical support from the hardware manufacturer, when trying to choose appropriate hardware. It may even require obtaining an evaluation sample for actual testing. TenAsys provides a tool, VtProbe, which checks a given platform for VT-x and VT-d support, see Appendix D.

3.2.5 Performance: How eVM addresses RTOS / Windows Application Needs

As we said earlier, the goal of eVM is to provide a virtual environment, hosted on a Windows platform, that looks and performs like a dedicated PC for the guest OS. The following discusses the performance issues to be aware of.

To provide bounded, real-time performance for the guest OS, the design of eVM is such that interrupts are delivered to the guest RTOS in order to be handled in a deterministic way. This is accomplished, in part, by making sure that real-time interrupts are not handled by Windows, and Windows is prevented from ever masking a real-time interrupt. The other part of deterministic interrupt handling is the design of the interrupt forwarding mechanism which is optimized to reduce interrupt latency and jitter and to provide a direct handoff to the real-time guest OS.

Hardware performance of the guest OS is maximized by providing a dedicated hardware thread for the guest OS. The guest does not have to share that core or hardware thread with any other operating system.

Since the guest OS has a dedicated hardware thread, Windows has its own dedicated hardware threads. Thus, Windows performance is not reduced by having to share CPU resources. Windows interrupts are handled directly by Windows. Aside from removing one core and some memory from the Windows system, there is minimal impact to Windows normal operation.

3.2.6 Communications between Windows and the Guest: Virtual Device Interfaces

When discussing the performance aspects of eVM, the isolation between Windows and the guest OS has been emphasized, but there will be times when both OSes will want to communicate with one another and share data. With both OSes on the same hardware, there are some unique opportunities to provide efficient communication mechanisms between the OSes, for example, there is a Virtual Ethernet connection that provides communication between the guest OS and Windows. This has an NDIS driver on the Windows side and an NE2000 interface on the guest OS side. This virtual interface uses shared memory to move packets between Windows and the guest OS as if there were a wired Ethernet link between the two OSes. There are also virtual serial ports that provide data interchange between Windows and the guest OS via a standard serial port interface.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

3.2.7 Communications between Windows and the Guest: Shared Memory

Windows and the guest OS can communicate via a block of shared memory. The shared memory is allocated from Windows non-paged pool and given to eVM to manage. Part or all of the memory can be mapped to a Windows process while access from the guest application is via a virtual PCI device which exposes the memory through a BAR (Base Address Register) in the PCI. See section 4.7 for further setup and usage details.

3.2.8 eVM Architecture: Partitioning the PC between Windows and Guest OS

Let's take a look at how the PC platform is partition using eVM to separate Windows and a guest OS.

3.2.8.1 Hardware Threads

On installation, one hardware thread is removed from Windows control and dedicated to the Guest OS. This uses standard Windows configuration mechanisms to manage this. In a multicore platform, this will dedicate one core to eVM and the guest OS.

3.2.8.2 Memory

On installation, the memory required for eVM and the guest OS is removed from the Windows memory pool and dedicated to the guest OS using similar standard Windows configuration mechanisms as for the thread control. This memory will be non-paged, because it's not under Windows control anymore. The memory that is dedicated to eVM and the guest OS is allocated before Windows boots.

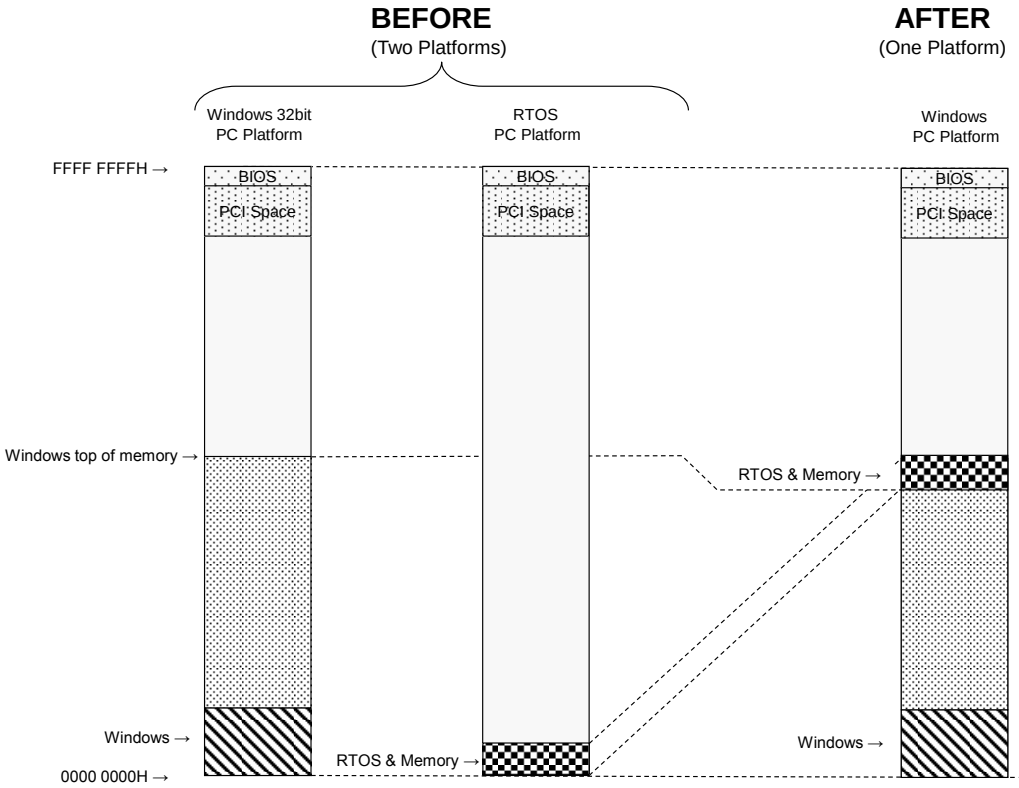


Figure 3.1 Memory map for a 32 bit installation

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

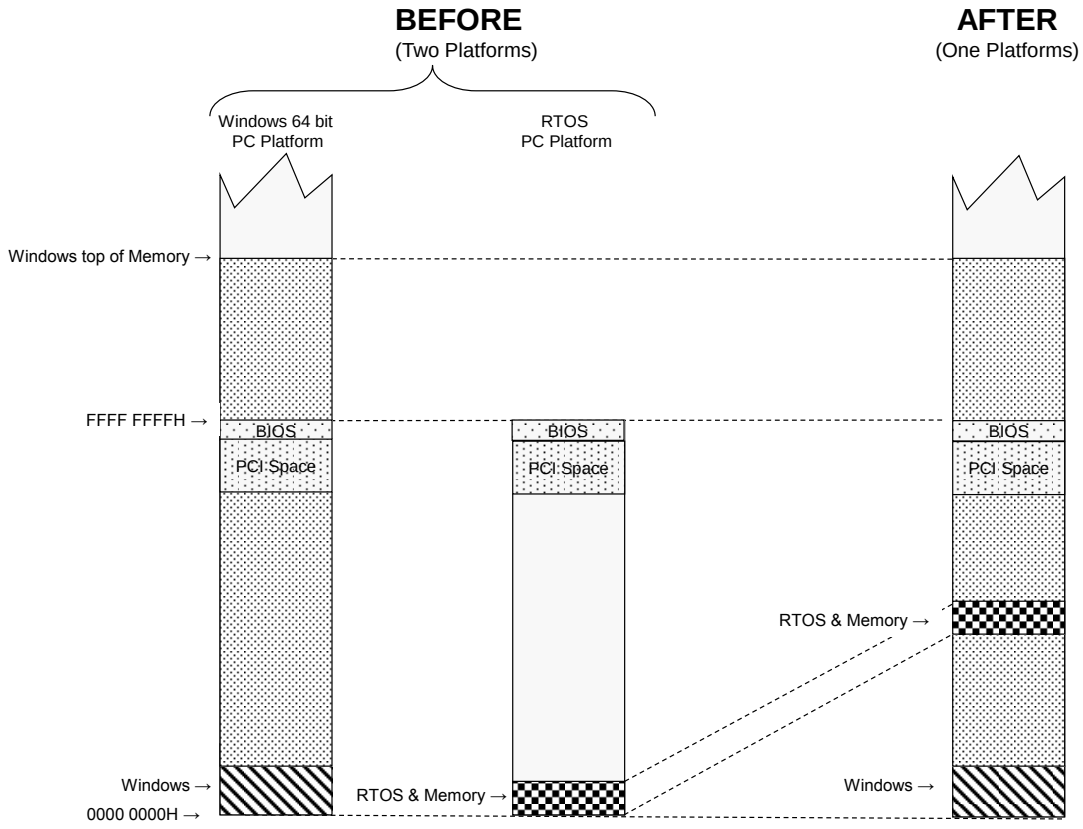


Figure 3.2 Memory map for a 64 bit installation

All memory addresses have to be translated to appear to be addressed from address 0 of the virtual PC, so it looks like a real dedicated PC. VT-x assists this memory remapping in hardware, thus there is no modification of the guest OS required.

3.2.8.3 Interrupts

In order to access a device directly from the guest OS, it is necessary to dedicate any interrupt resource associated with that device to eVM, so eVM can manage it and not Windows. eVM maintains the control of these resources and then passes control to the Guest OS. With eVM, not Windows managing these interrupt resources, the guest OS maintains a deterministic system environment.

3.2.8.4 Devices

Virtual device is one class of device: disc controllers, CMOS, PCI bus, host bridge on the PCI bus, timers, and interrupt controller. These devices are created in software and presented to the guest OS as if they were real hardware devices. These virtual devices are all part of the virtual machine.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

A second class of devices is of real devices that are assigned to the guest OS and taken away from Windows. The guest OS has exclusive access to these devices and does not share them with Windows. A dedicated PCIe Ethernet card is an example of this. PCI or PCIe devices can be dedicated to the guest OS such that an application specific device is managed by the RTOS and Windows is not aware of the device.

3.2.9 Resources that are not Isolated between Windows and the Guest OS

Per the virtualization architecture, eVM provides isolation between Windows and the Quest OS, but there are shared resources that cannot be managed in software and the usage of these resources by one of the OSes can impact the other OS. The memory controller, internal graphics controller (also uses the memory controller), and the front side bus are examples of these kinds of resources. For example, heavy usage of the memory controller by one of the OSes is going to impact performance on the whole platform. The selection of processor, chipset, and clocking will have an influence on the overall system performance. For the very highest performance applications, the choice of hardware is going to have to be done carefully and evaluated with the appropriate measurements to determine the optimum hardware platform.

3.2.10 Device Management

The eVM Device Manager utility is used for the management of devices that will be dedicated to that guest OS. Before using the eVM Device Manager to manage devices, it is important to know what devices can be dedicated to the guest OS, what devices cannot, and why.

3.2.10.1 What is a device

The devices that eVM can dedicate to a guest OS are PCI devices, PCI Express (PCIe) devices, and legacy devices (COM, LPT, etc.). These devices come in a variety of flavors; so it is important to determine if the device can, in fact, be dedicated to a guest OS. The hardware features that will be used to determine the device's suitability for being a dedicated, guest OS, device are:

1. Device Access: Does the device use memory mapped registers or I/O mapped registers?
2. Interrupt capability: Does the device generate an interrupt; and if so, what type of interrupt does it generate?
3. Bus Master capability: Is the device a bus master device, i.e. does the device generate its own memory accesses?

An example of a device that cannot be dedicated to the guest OS would be a USB memory disk. That is because the USB memory disk interfaces through the USB controller. It is not interfaced directly to the bus nor is it accessed directly by the device driver.

3.2.10.2 Identifying your device's capabilities

The following describes the 3 classes of device capability that are of interest when configuring devices in eVM for dedicated access by the guest OS.

1. Device Access:

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

The method of device access to the device registers must be determined. The device can either use the I/O space or be memory mapped. The CPU Virtualization Technology, VT-x, can manage the mapping of direct memory or I/O space for the virtual machine. The VT-x mapping also provides memory protection. Any access to a memory or I/O space that is not mapped would be trapped by eVM and blocked. Thus errant driver software in the guest OS could not corrupt the Windows platform.

Legacy devices must be explicitly configured. Their I/O ranges will have to be determined and configured in eVM, so that eVM can allow access to these I/O ports by the guest OS.

PCI and PCIe devices can be discovered. All the PCI/PCIe register and memory access information is contained in the PCI header. eVM reads the Windows configuration for the device and sets up the device's registers and memory to be transparently accessible from the guest OS. No interaction is necessary by the user to configure this device after it has been assigned to eVM.

2. Interrupt Capability:

The interrupt capability of the device needs to be determined. If the device has interrupt capability, then it needs to be determined if the interrupt will be used by the guest OS. eVM then needs to be appropriately configured.

3. Bus Master Capability

Finally, it needs to be determined if the device is a bus master device, i.e. does the device contain a DMA engine and can it generate its own memory accesses.

3.2.10.3 Interrupts

There are two classes of interrupts. The first class is the legacy interrupts, or IRQs that have the device interrupt lines connected to specific inputs of an interrupt controller, and the interrupt controller manages how each interrupt, IRQ, is delivered to the CPU. The second class is the more modern Message Signaled Interrupt (MSI), where the device performs a special bus write to signal the chipset. The device has no interrupt lines. Instead it has special parameters that the device writes to a memory space reserved for MSI interrupt activity. When the device writes to the MSI memory space, the interrupt is delivered directly to the CPU.

eVM will handle both classes of interrupts for devices that are dedicated to the guest OS. All dedicated device interrupts are fielded by eVM. Then eVM, in turn, forwards those interrupts to the guest OS when it is ready to receive interrupts. The virtual machine provides virtual interrupt services for the guest OS. The guest OS may mask and unmask interrupts in the virtual machine. eVM monitors this, and holds off interrupt forwarding for interrupts that are masked in the virtual machine, and then forwards those interrupts to the virtual machine when they are unmasked.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

3.2.10.4 Software Considerations

To provide a PC-like virtual machine, eVM must manage both classes of interrupt and act as an interrupt controller for the virtual machine. The guest OS interacts with the virtual interrupt controller of eVM as if it were a hardware interrupt controller. The eVM virtual interrupt controller provides interrupt signaling to the guest OS, interrupt masking capability, and interrupt enabling and disabling. Therefore, if the guest has interrupts disabled, then eVM must wait until the interrupts are enabled before passing the interrupt to the guest OS.

The virtual machine that supports the guest OS has a different interrupt mapping for the virtual machine than the actual hardware platform. The virtual machine presents an 8259 legacy interrupt controller as the means for interfacing the dedicated device interrupts to the guest OS. eVM maps both classes of interrupts, legacy and MSI, as IRQ's in the virtual machine. The guest OS sees all interrupts as legacy IRQ interrupts in the virtual machine irrespective of whether the real hardware interrupt was a legacy interrupt or an MSI interrupt. eVM is acting like a software interrupt controller. Specific mapping of devices to IRQs in the virtual machine is described by the Virtual Machine specification, see Appendix B.

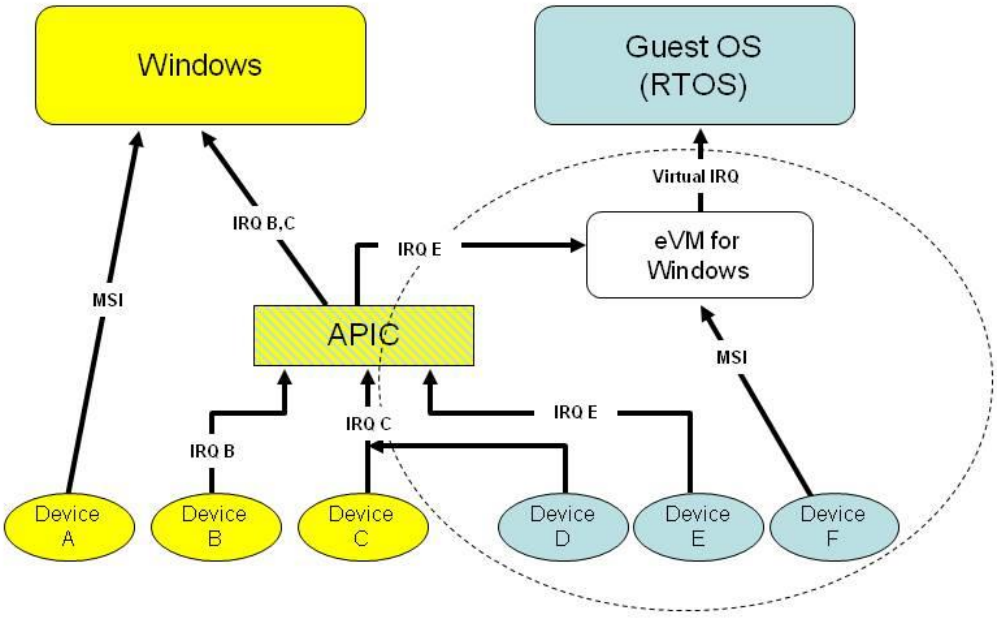


Figure 3.3 Interrupt management

Device A: Is a PCI device and the driver supports MSI – ***Note: Windows XP and earlier versions of Windows do not support MSI***.
Device B: Is handled normally by Windows.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

Device C and D: Devices share the same IRQ line and the RTOS needs to access the Device D, so the device is passed to the RTOS but not the IRQ. Device C remains a fully functional Windows device. Device D may be polled by the Guest OS, but it cannot deliver an interrupt to the guest OS.
Device E: Is passed on to the guest OS. IRQ E is serviced by eVM then delivered to the RTOS as a virtual IRQ.
Device F: Is an MSI device. The MSI is handled by eVM and delivered to the RTOS as a virtual IRQ.
Note: eVM always handles the IRQ, meaning that eVM's interrupt handler runs when the interrupt fires. The eVM interrupt handler converts the interrupt to an input for the virtual Peripheral Interrupt Controller (PIC) and forwards the translated vector to the guest OS.

Example: Ethernet controller capture
If an Ethernet controller that uses IRQ18 is to be captured and dedicated to the virtual machine, and hence, the guest OS, the first thing that eVM will do is install an interrupt handler for IRQ18 to manage the real interrupt on the hardware. Next, eVM will map that interrupt into one of the virtual machine's IRQs, a different IRQ from the actual hardware interrupt. Then in operation, the eVM interrupt handler will handle IRQ18. It will map IRQ18 to an available IRQ on the virtual machine, and it will forward the interrupt to the guest OS when the guest OS is ready to handle it.

There is some additional latency introduced by the forwarding and mapping of the real hardware interrupt to the interrupt on the virtual machine, but this latency is fixed and bounded. Because this latency is fixed and bounded, the guest OS can still run in a deterministic fashion. eVM has been optimized to minimize this latency, but this latency is also determined by the hardware. Some CPU, chipset, and board designs are faster than others, but the hardware contribution to this latency will be fixed. The virtual machine will still be deterministic.

3.2.10.5 System hardware considerations – what eVM can or can't do!

There are some restrictions to what can and cannot be done when mapping interrupts to the virtual machine and still maintain a deterministic system. Legacy interrupts from PCI devices present a restriction, because multiple PCI devices can share a single interrupt. In fact, PCI is designed to share an interrupt. This interrupt sharing presents a problem, if the devices that are sharing an interrupt are not configured to be used by the same OS. That is to say, if one device is to be used as a Windows device and another device is to be used as a guest OS device and they both share the same interrupt, then there is no way to provide deterministic response for the guest device, since it is dependent on a Windows interrupt handler running.

When an interrupt occurs on a shared interrupt, the interrupt handlers for all devices sharing that interrupt must be run to determine which device caused the interrupt and to clear the interrupt so it will be available to handle a subsequent event. This would mean that the guest

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

OS would be dependent on a Windows interrupt handler running to maintain a deterministic system. Therefore, it is not allowed to assign a device to the guest and a device to Windows that share a common interrupt. The eVM Device Manager will flag an attempt to do this as a conflict. A conflict of this type is a property of the motherboard and/or chipset design. There is no way to configure around that, so it is important to choose a hardware design that will allow interrupt isolation of devices that are to be dedicated to the guest OS. One alternative for a device that would have an interrupt conflict with another device used by Windows would be to run the device in a polled mode. The device would be configured with interrupts turned off and register access can still be dedicated to the guest OS. It should be noted that MSI interrupts do not have this problem.

3.2.10.6 DMA

A DMA device can generate its own memory accesses without CPU intervention. Such a device is a bus master device. For bus master devices, the Intel Virtualization Technology, VT-x, plays no part. VT-x manages address remapping for the CPU. Addresses that the DMA engine of a bus master device would use must be remapped for the virtual machine. The view of the memory that is assigned to the virtual machine starts at address zero, but the actual physical address of that memory is offset from memory address zero an amount dependent on the system configuration. Any addresses that the DMA engine would use must be mapped to the actual physical address of the memory, so the physical memory offset must be added to the virtual memory addresses of the virtual machine.

This DMA address mapping can be handled in two ways. One way is to modify the device driver so that it will do the DMA address mapping. This requires that a modified device driver must be prepared and installed on the guest OS. The modified device driver must be able to determine the memory offset of the virtual machine's physical memory and apply that offset to any DMA operation, see Appendix E. The other method for DMA address mapping is to use hardware that is specifically designed to provide address mapping of bus master devices. Intel Virtualization Technology, VT-d or Virtualization for Directed I/O, is the hardware mechanism that is available for DMA address mapping. The VT-d technology resides in specific chipsets that contain a DMA redirection unit that translates bus master address accesses on the fly.

3.2.10.7 Hardware consideration – VT-d

eVM defines an address domain that is assigned to the virtual machine for the guest OS. Any device assigned to eVM is also assigned to that memory domain in hardware. eVM will then translate any address that is generated in that domain and map it to the physical address of the virtual machine domain. If a device tries to access memory outside its domain, it is blocked by eVM, providing memory security and hardware protection, as well as, memory mapping.

If VT-d is not present, a standard bus master driver will not work and will compromise Windows, because a bus master device that is operating with a device driver that has not been modified for the virtual machine, will address the wrong area of memory, and almost certainly corrupt

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

Windows memory. The bus master driver must be modified to work with the virtual machine and do the appropriate address translations to be able to function in the virtual machine if VT-d is not present. Therefore, it is highly desirable to have VT-d supported by the hardware and enabled through the BIOS, if bus master devices are to be dedicated to the virtual machine, and hence, the guest OS.

3.2.11 What Happens when a Device is assigned to eVM

When the eVM Device Manager is used to assign a device to eVM from Windows, the following actions are taken:

- Access to the device by Windows is stopped by removing the Windows device driver for that device.
- Device identity information is passed from Windows to eVM, so eVM can recognize it. Thus when eVM next starts up, it gets notified that it now owns this device.
- Device identity can be passed to eVM with its IRQ or without its IRQ. If the device is passed with its IRQ, then a check is made for an interrupt sharing violation. If there is an interrupt sharing violation, the eVM Device Manager will post a warning.
- When eVM next starts and sees that this device now belongs to eVM, it obtains the device identifier and discovers the device. If the device has an IRQ, eVM will assign an interrupt handler for it. If it is PCI device, eVM inserts the device into one of the virtual PCI slots that are provided by the virtual machine. The virtual machine that eVM creates has 5 virtual PCI slots for dedicated PCI devices.
- If the device is a PCI device, eVM Scans the base address registers for that device and configures the virtual machine so the access to those base address registers go directly to that device and are not intercepted by eVM.
- If the device has an interrupt, eVM assigns a virtual IRQ in the virtual machine's interrupt map.
- If the device does not have an IRQ, eVM will disable the interrupt in the virtual PCI device header of the virtual machine.
- If this device is capable of being a bus master and if VT-d is enabled, then eVM assigns the device to the VT-d domain of the virtual machine for hardware address translation in the guest OS.
- If this device is capable of being a bus master but VT-d is not present, a warning will be presented by the eVM Device Manager. It is the responsibility of the developer to either provide the modified driver or disable bus master capability.

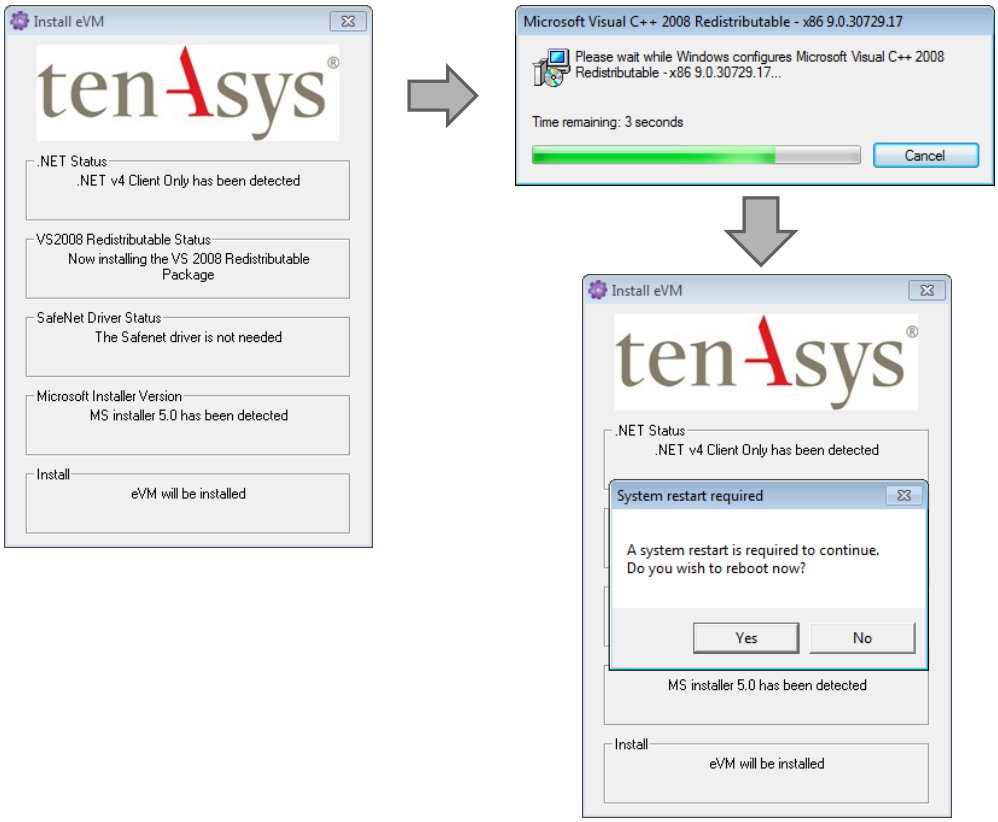
Copyright © 2014 TenAsys Corporation,
All Rights Reserved

4 Installation

4.1 Installation of eVM for Windows

The following provides a step-by-step guide of the installation screens with comments as necessary, including Windows and eVM setup and License activation. The following description is for installation on Windows 8/7/Vista.

Run **evm20_installer.exe** which begins the installation process by installing the Microsoft Visual C++ 2008 Redistributable. Installer first extracts files to temporary folder, detects what applications are loaded on the system and loads Microsoft Visual C++ Redistributable if they are not already installed.


Install eVM
.NET Status
.NET v4 Client Only has been detected
VS2008 Redistributable Status
Now installing the VS 2008 Redistributable Package
SafeNet Driver Status
The Safenet driver is not needed
Microsoft Installer Version
MS installer 5.0 has been detected
Install
eVM will be installed
Microsoft Visual C++ 2008 Redistributable - x86 9.0.30729.17
Please wait while Windows configures Microsoft Visual C++ 2008 Redistributable - x86 9.0.30729.17...
Time remaining: 3 seconds
Cancel
Install eVM
.NET Status
.NET v4 Client Only has been detected
System restart required
A system restart is required to continue.
Do you wish to reboot now?
Yes
No
MS installer 5.0 has been detected
Install
eVM will be installed

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

1. You will get this message if and **only** if your computer does not support VT-x technology. See Appendix B1 – System Requirement for more information. Otherwise screen on shown in Figure 4.4 will come up

eVM for Windows Setup
eVM®
for Windows
tenAsys
Welcome to the eVM for Windows Installation Wizard
eVM has detected that VT-x technology is not present on this system. VT-x is required to install eVM.
eVM has detected that VT-d technology is either not present on this system, or not enabled. You will not be able to use the full functionality of eVM on this platform in its current configuraiton.
Click 'I Understand' below to continue.
< Back
Next >
Cancel

Figure 4.1 VT-x not supported message

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

2. With the Visual C++ 2008 Redistributable installation complete, the eVM Installation Wizard is automatically launched, see Figure 4.2. Click **I Understand** after reading the notice and then **Next** to continue.

eVM for Windows Setup
eVM®
for Windows®
tenAsys®
Welcome to the eVM for Windows Installation Wizard
It is strongly recommended that you exit all Windows programs before running this setup program.
Click Cancel to quit the setup program, then close any programs you have running. Click Next to continue the installation.
WARNING: This program is protected by copyright law and international treaties.
eVM has detected that VT-d technology is either not present on this system, or not enabled. You will not be able to use the full functionality of eVM on this platform in its current configuraiton.
Click 'I Understand' below to continue.
I Understand
< Back
Next >
Cancel

Figure 4.2 Installation Wizard

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

3. Read and accept the TenAsys eVM Software License Agreement; and click **Next** to continue, see Figure 4.3.Read Figure 4.4 Release Notes and select *Next* to continue with the installation.

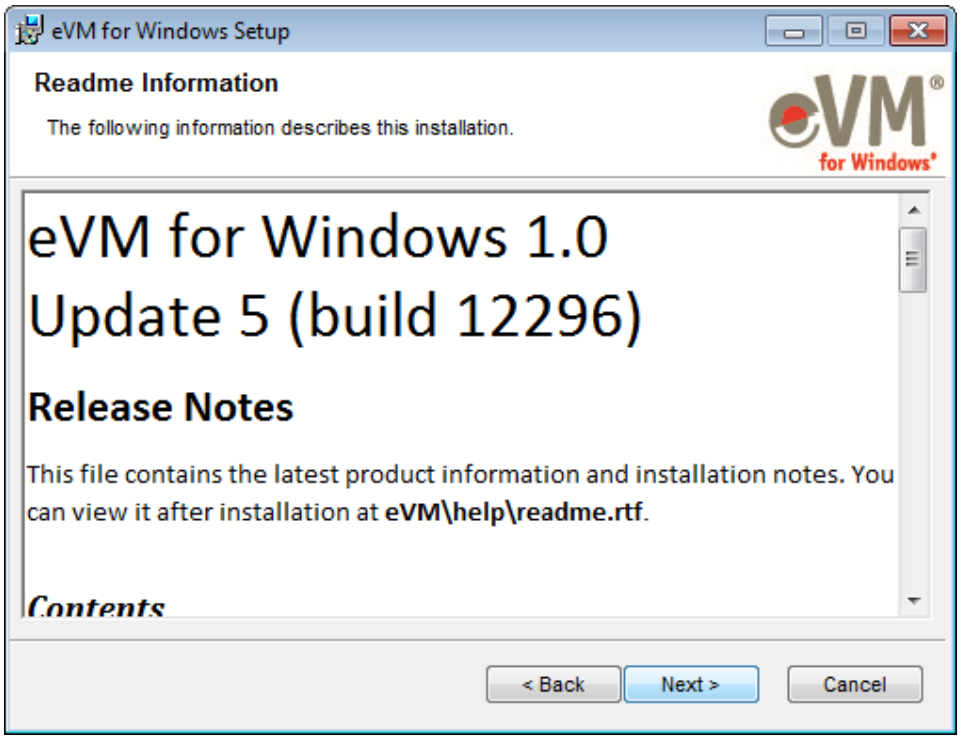

eVM for Windows Setup
Readme Information
The following information describes this installation.
eVM for Windows 1.0
Update 5 (build 12296)
Release Notes
This file contains the latest product information and installation notes. You can view it after installation at eVM\help\readme.rtf.
Contents
< Back
Next >
Cancel

Figure 4.4 Release Notes

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

4.

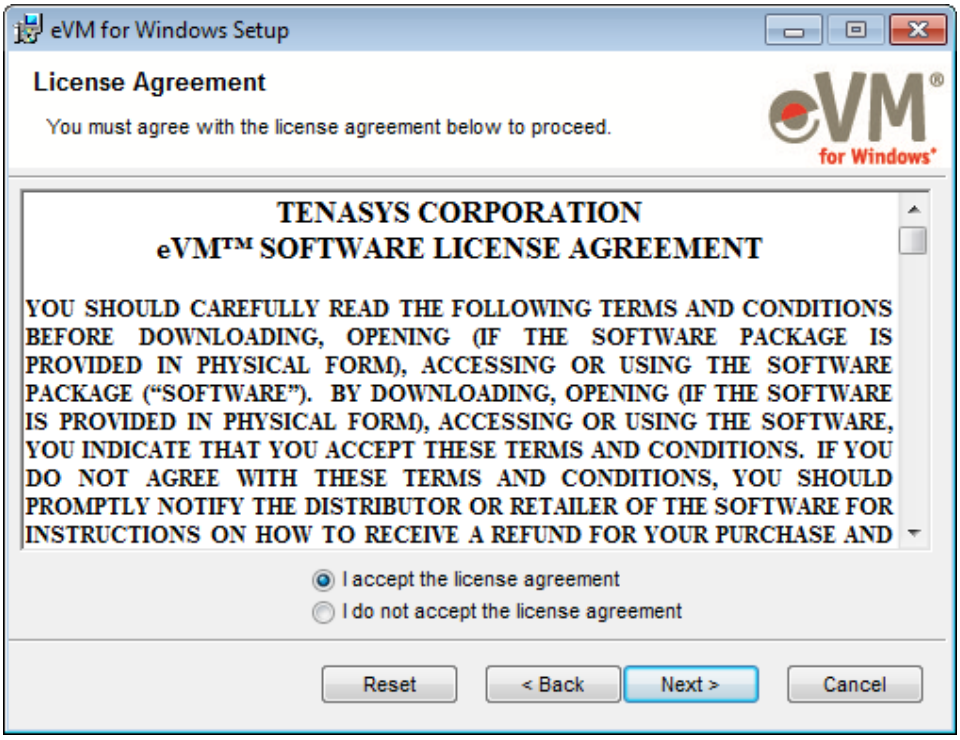

eVM for Windows Setup
License Agreement
You must agree with the license agreement below to proceed.
eVM®
for Windows*
TENASYS CORPORATION
eVM™ SOFTWARE LICENSE AGREEMENT
YOU SHOULD CAREFULLY READ THE FOLLOWING TERMS AND CONDITIONS BEFORE DOWNLOADING, OPENING (IF THE SOFTWARE PACKAGE IS PROVIDED IN PHYSICAL FORM), ACCESSING OR USING THE SOFTWARE PACKAGE ("SOFTWARE"). BY DOWNLOADING, OPENING (IF THE SOFTWARE IS PROVIDED IN PHYSICAL FORM), ACCESSING OR USING THE SOFTWARE, YOU INDICATE THAT YOU ACCEPT THESE TERMS AND CONDITIONS. IF YOU DO NOT AGREE WITH THESE TERMS AND CONDITIONS, YOU SHOULD PROMPTLY NOTIFY THE DISTRIBUTOR OR RETAILER OF THE SOFTWARE FOR INSTRUCTIONS ON HOW TO RECEIVE A REFUND FOR YOUR PURCHASE AND
I accept the license agreement
I do not accept the license agreement
Reset
< Back
Next >
Cancel

Figure 4.3 License Agreement

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

5. Read Figure 4.4 Release Notes and select *Next* to continue with the installation.

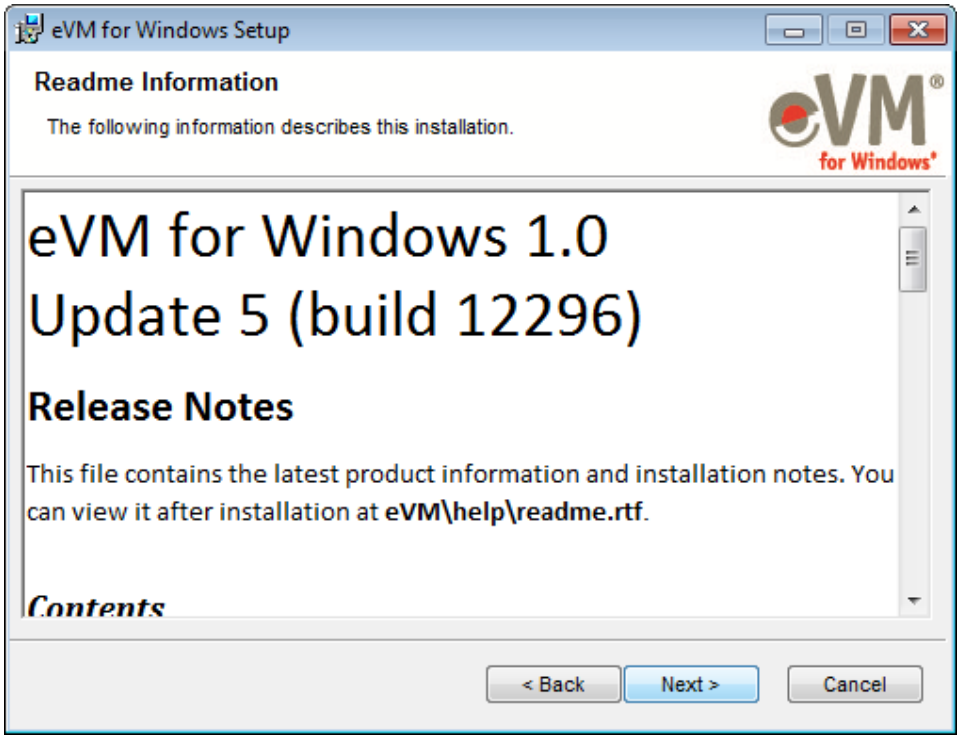

eVM for Windows Setup
Readme Information
The following information describes this installation.
eVM for Windows 1.0
Update 5 (build 12296)
Release Notes
This file contains the latest product information and installation notes. You can view it after installation at eVM\help\readme.rtf.
Contents
< Back
Next >
Cancel

Figure 4.4 Release Notes

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

6. Select the folder where you would like the installer to put the eVM installation files. The default location is “C:\Program Files\eVM\”, see Figure 4.5. Then click **Next** to continue.

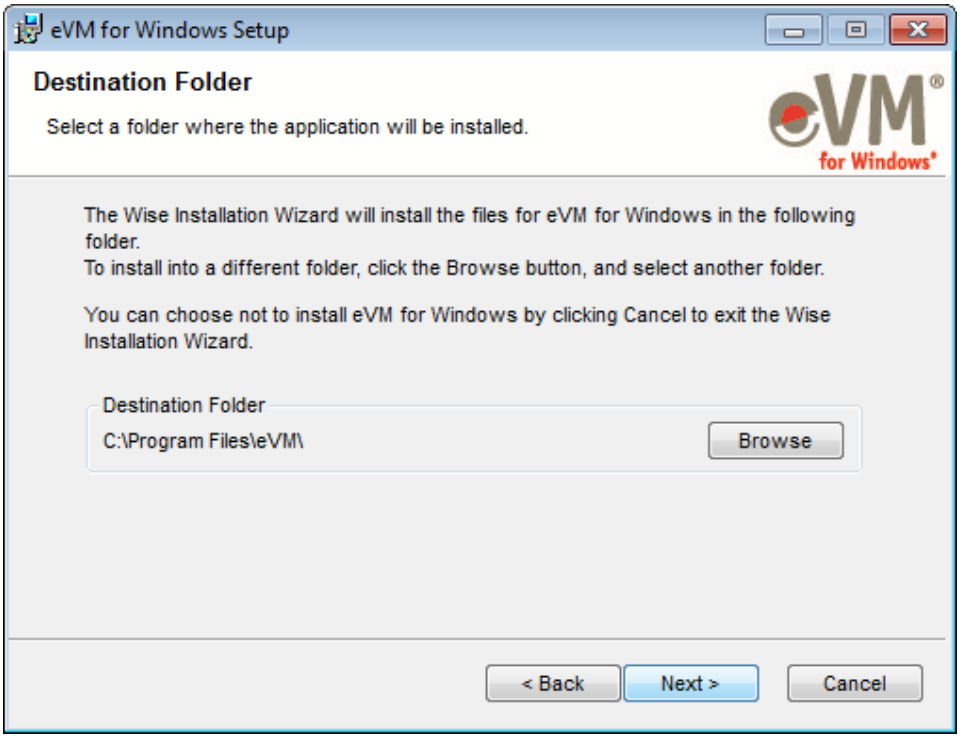

eVM for Windows Setup
Destination Folder
Select a folder where the application will be installed.
eVM
for Windows
The Wise Installation Wizard will install the files for eVM for Windows in the following folder.
To install into a different folder, click the Browse button, and select another folder.
You can choose not to install eVM for Windows by clicking Cancel to exit the Wise Installation Wizard.
Destination Folder
C:\Program Files\eVM\
Browse
< Back
Next >
Cancel

Figure 4.5 - Select eVM Installation Folder

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

7. Select the type of installation that you would like. If you choose *Complete Installation*, Figure 4.6, eVM with virtual port support, all the utilities, and a sample guest OS, iRMX will be installed. You can also choose a custom installation. After selecting the installation type, click **Next** to continue.

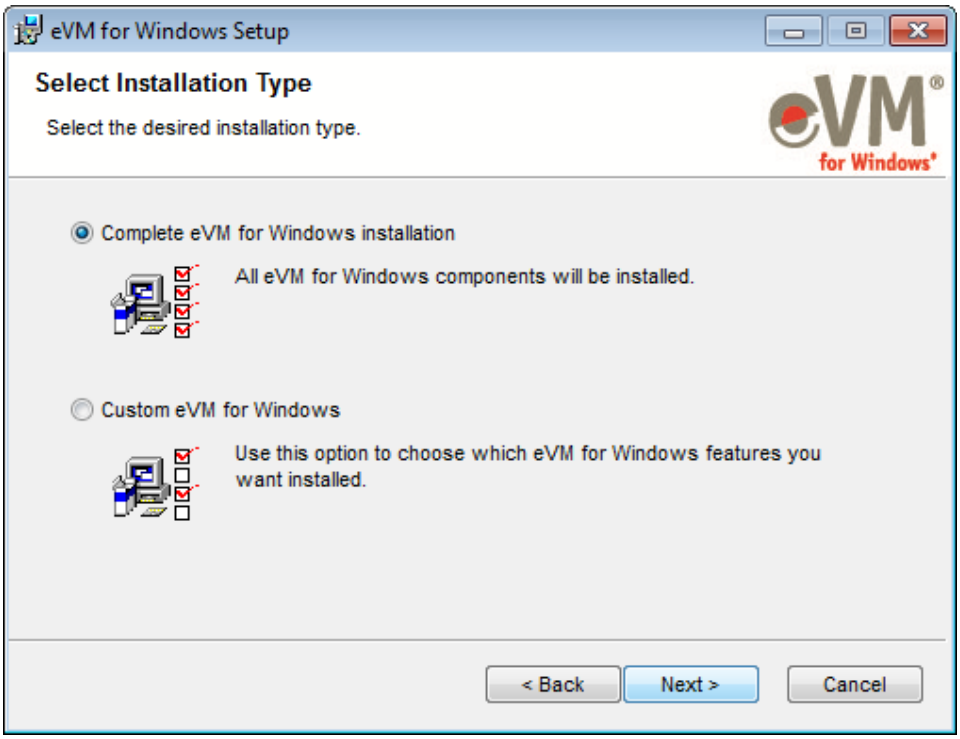

eVM for Windows Setup
Select Installation Type
Select the desired installation type.
eVM
for Windows
Complete eVM for Windows installation
All eVM for Windows components will be installed.
Custom eVM for Windows
Use this option to choose which eVM for Windows features you want installed.
< Back
Next >
Cancel

Figure 4.6 - Choosing Standard Installation Type

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

8. The eVM Installation Wizard will now prompt you to enter your name, organization, and your eVM activation code that you received when you purchased and downloaded eVM, see Figure 4.7. Make sure you fill out all the fields properly and **be sure that you are connected to the Internet, as the activation code will be verified with TenAsys over the Internet**. Then Click **Next** to proceed with the installation.

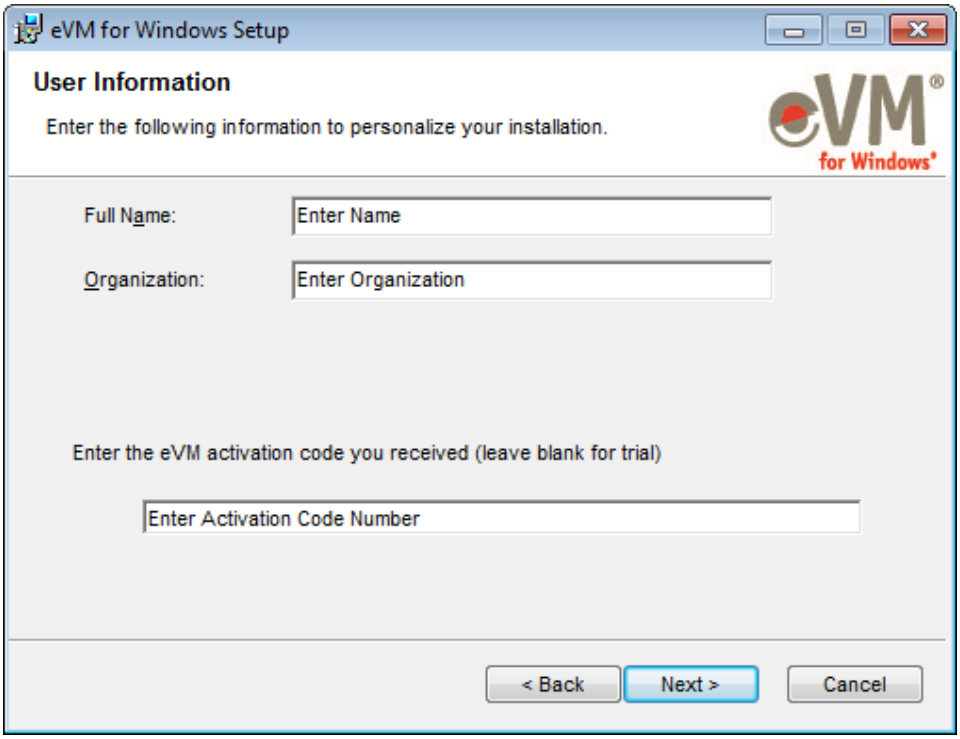


Figure 4.7 - eVM Activation

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

9. When your activation code has been verified, the eVM Installation Wizard will then prompt you to confirm that you want to proceed with the installation, Figure 4.8. If you want to check or change anything at this point, you can click the **Back** button.

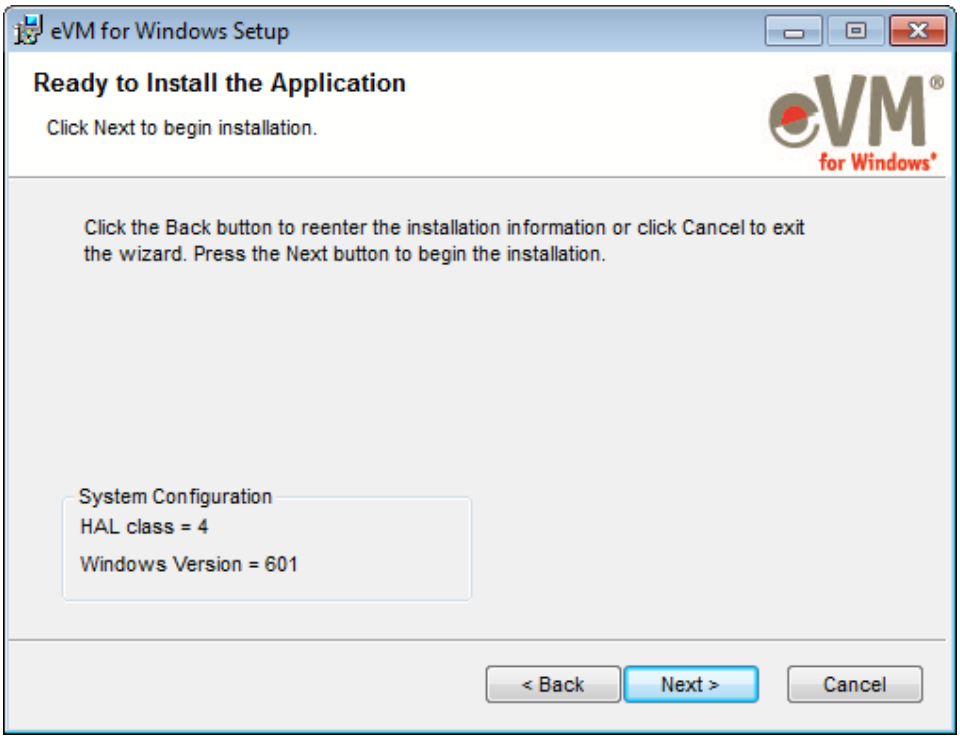

eVM for Windows Setup
Ready to Install the Application
Click Next to begin installation.
eVM®
for Windows®
Click the Back button to reenter the installation information or click Cancel to exit the wizard. Press the Next button to begin the installation.
System Configuration
HAL class = 4
Windows Version = 601
< Back
Next >
Cancel

Figure 4.8 – eVM Installation Verification

To proceed with the installation, click **Next**.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

10. You will then see a progress dialog, see Figure 4.9, the will show you the status of the components and features that are being installed.

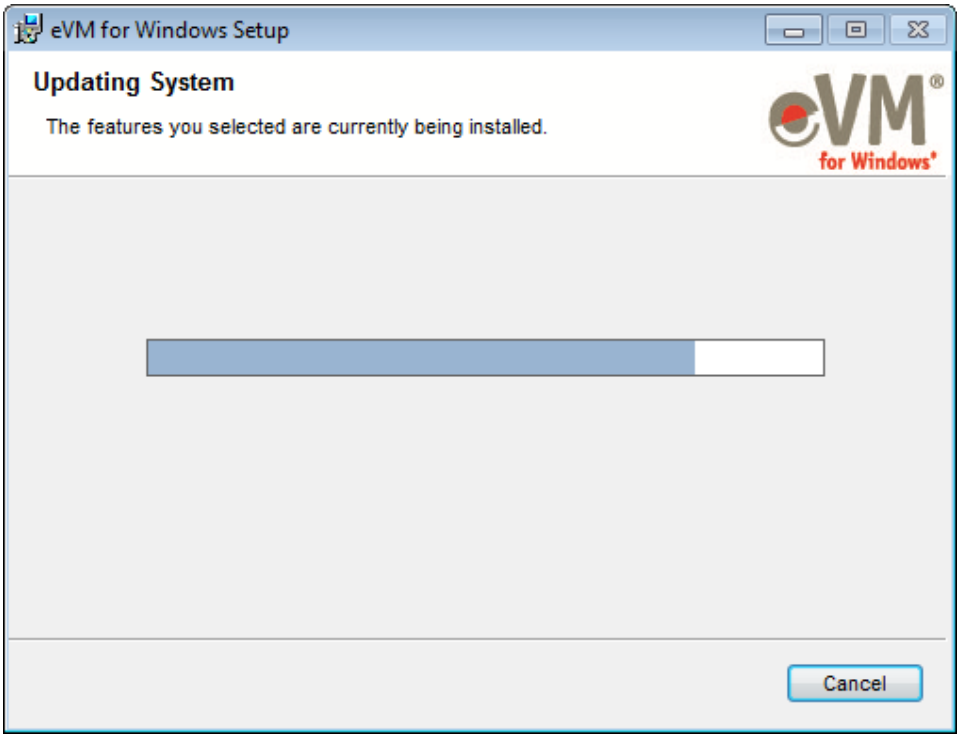


Figure 4.9 – eVM Installation Wizard Progress Dialog

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

11. When all components and features of eVM that you selected have been installed, the Wizard will indicate that the installation was completed successfully, see Figure 4.10. If you want to view the installation log to see what the Wizard did, check the **checkbox** to view the log. Whether you view the log or not, note the path where the log is saved. If you experience difficulties with the installation that you cannot resolve, TenAsys technical support may ask to see this log to help determine the problem. Now click **Finish**.

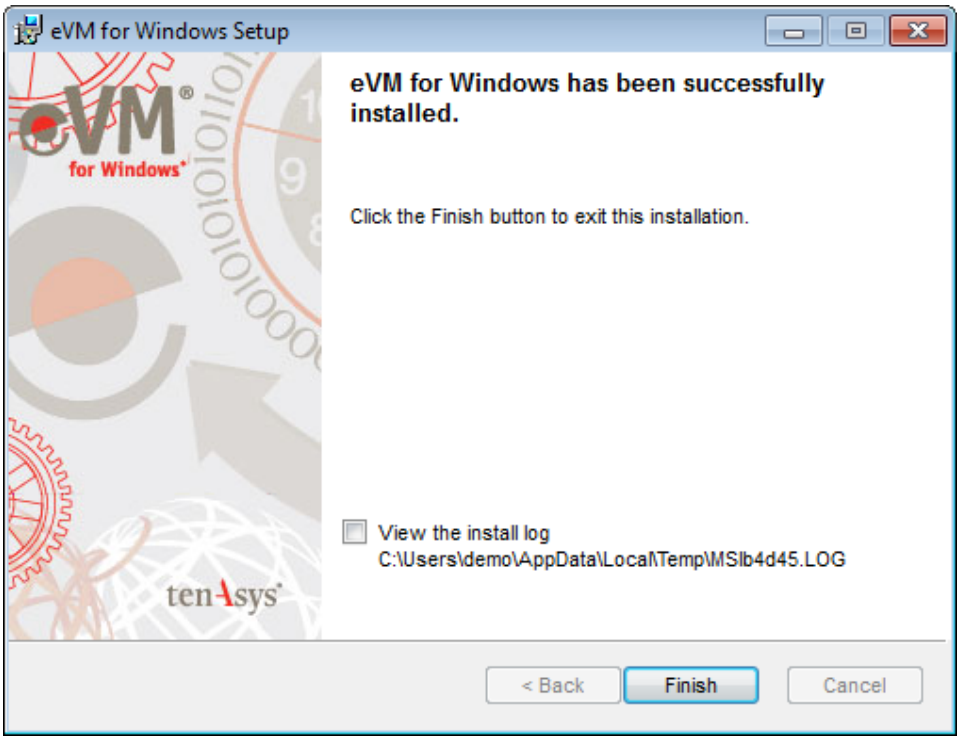


Figure 4.10 - eVM Installation Wizard Successful Installation

The first time eVM for Windows is installed on the target system; the installer will load TenAsys drivers to support the virtual serial port. Prior to this operation you will be prompted with a screen as shown in Figure 4.11 Note: If you don't check the box "Always trust software from "TenAsys Corporation", this prompt will come up again when you install additional virtual ports.

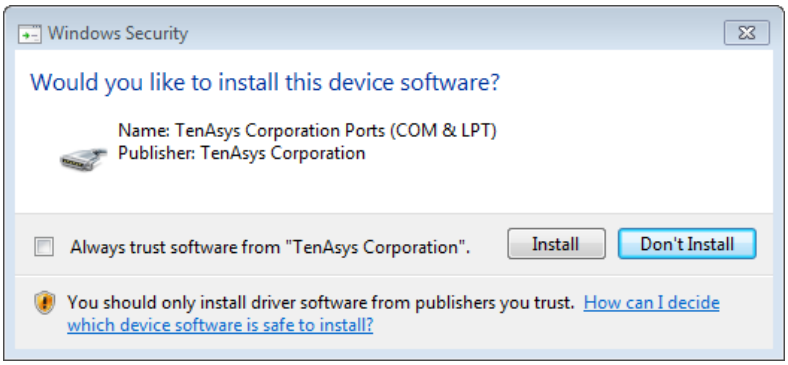

Windows Security
Would you like to install this device software?
Name: TenAsys Corporation Ports (COM & LPT)
Publisher: TenAsys Corporation
Always trust software from "TenAsys Corporation".
Install
Don't Install
You should only install driver software from publishers you trust. How can I decide which device software is safe to install?

Figure 4.11 TenAsys Driver loading notification

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

12. If you chose to view the installation log, you will see something similar to Figure 4.12. After viewing the log, you can close Notepad.

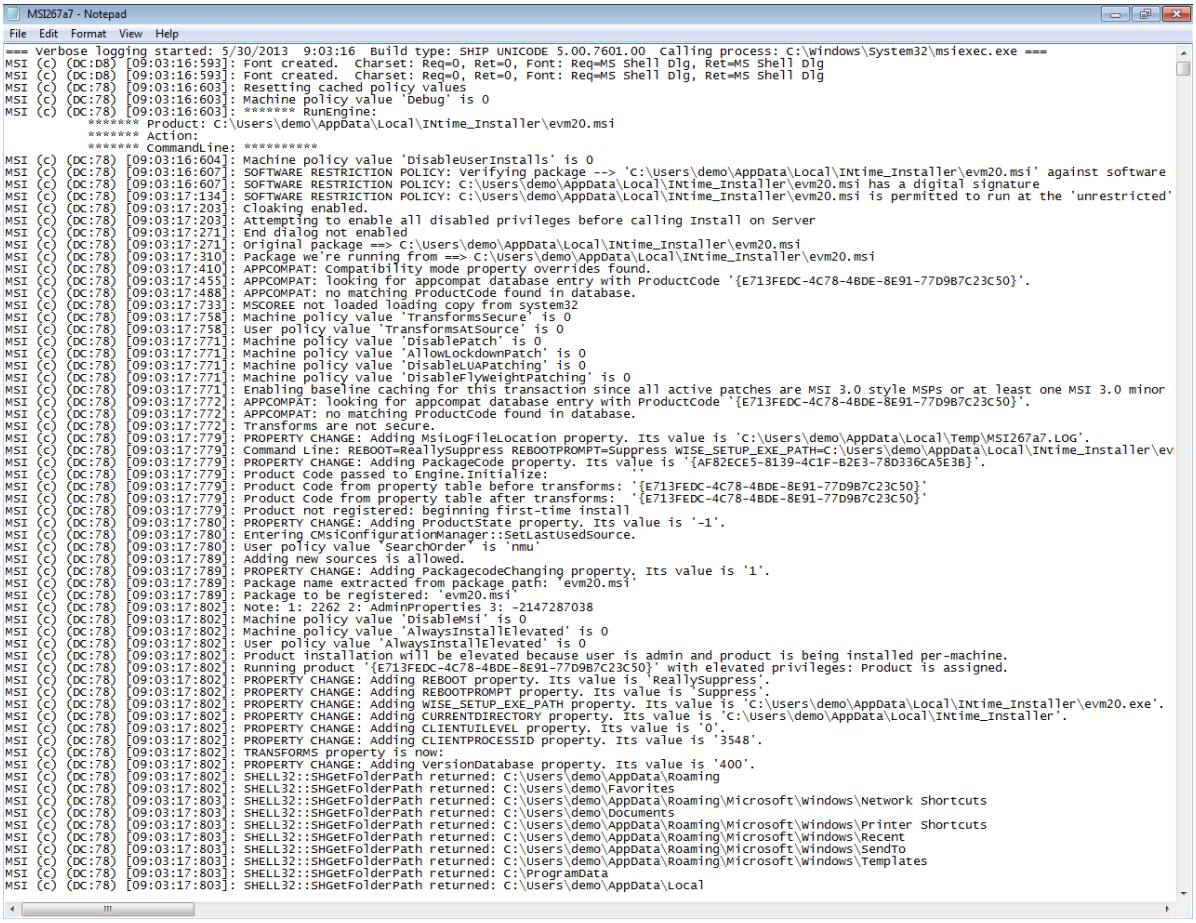

MSI267a7 - Notepad
File Edit Format View Help
=== Verbose logging started: 5/30/2013 9:03:16 Build type: SHIP UNICODE 5.00.7601.00 Calling process: C:\Windows\System32\msiexec.exe ===
MSI (c) (DC:D8) [09:03:16:593]: Font created. Charset: Req=0, Ret=0, Font: Req=MS Shell Dlg, Ret=MS Shell Dlg
MSI (c) (DC:D8) [09:03:16:593]: Font created. Charset: Req=0, Ret=0, Font: Req=MS Shell Dlg, Ret=MS Shell Dlg
MSI (c) (DC:78) [09:03:16:603]: Resetting cached policy values
MSI (c) (DC:78) [09:03:16:603]: Machine policy value 'Debug' is 0
MSI (c) (DC:78) [09:03:16:603]: ******* RunEngine:
******* Product: C:\Users\demo\AppData\Local\INtime_Installer\evm20.msi
******* Action:
******* CommandLine: **********
MSI (c) (DC:78) [09:03:16:604]: Machine policy value 'DisableUserInstalls' is 0
MSI (c) (DC:78) [09:03:16:607]: SOFTWARE RESTRICTION POLICY: Verifying package --> 'C:\Users\demo\AppData\Local\INtime_Installer\evm20.msi' against software
MSI (c) (DC:78) [09:03:16:607]: SOFTWARE RESTRICTION POLICY: C:\Users\demo\AppData\Local\INtime_Installer\evm20.msi has a digital signature
MSI (c) (DC:78) [09:03:17:134]: SOFTWARE RESTRICTION POLICY: C:\Users\demo\AppData\Local\INtime_Installer\evm20.msi is permitted to run at the 'unrestricted'
MSI (c) (DC:78) [09:03:17:203]: Cloaking enabled.
MSI (c) (DC:78) [09:03:17:203]: Attempting to enable all disabled privileges before calling Install on Server
MSI (c) (DC:78) [09:03:17:271]: End dialog not enabled
MSI (c) (DC:78) [09:03:17:271]: Original package ==> C:\Users\demo\AppData\Local\INtime_Installer\evm20.msi
MSI (c) (DC:78) [09:03:17:310]: Package we're running from ==> C:\Users\demo\AppData\Local\INtime_Installer\evm20.msi
MSI (c) (DC:78) [09:03:17:410]: APPCOMPAT: Compatibility mode property overrides found.
MSI (c) (DC:78) [09:03:17:455]: APPCOMPAT: looking for appcompat database entry with ProductCode '{E713FEDC-4C78-4BDE-8E91-77D9B7C23C50}'.
MSI (c) (DC:78) [09:03:17:488]: APPCOMPAT: no matching ProductCode found in database.
MSI (c) (DC:78) [09:03:17:733]: MSCOREE not loaded loading copy from system32
MSI (c) (DC:78) [09:03:17:758]: Machine policy value 'TransformsSecure' is 0
MSI (c) (DC:78) [09:03:17:758]: User policy value 'TransformsAtSource' is 0
MSI (c) (DC:78) [09:03:17:771]: Machine policy value 'DisablePatch' is 0
MSI (c) (DC:78) [09:03:17:771]: Machine policy value 'AllowLockdownPatch' is 0
MSI (c) (DC:78) [09:03:17:771]: Machine policy value 'DisableLUAPatching' is 0
MSI (c) (DC:78) [09:03:17:771]: Machine policy value 'DisableFlyWeightPatching' is 0
MSI (c) (DC:78) [09:03:17:771]: Enabling baseline caching for this transaction since all active patches are MSI 3.0 style MSPs or at least one MSI 3.0 minor
MSI (c) (DC:78) [09:03:17:772]: APPCOMPAT: looking for appcompat database entry with ProductCode '{E713FEDC-4C78-4BDE-8E91-77D9B7C23C50}'.
MSI (c) (DC:78) [09:03:17:772]: APPCOMPAT: no matching ProductCode found in database.
MSI (c) (DC:78) [09:03:17:772]: Transforms are not secure.
MSI (c) (DC:78) [09:03:17:779]: PROPERTY CHANGE: Adding MsiLogFileLocation property. Its value is 'C:\Users\demo\AppData\Local\Temp\MSI267a7.LOG'.
MSI (c) (DC:78) [09:03:17:779]: Command Line: REBOOT=ReallySuppress REBOOTPROMPT=Suppress WISE_SETUP_EXE_PATH=C:\Users\demo\AppData\Local\INtime_Installer\ev
MSI (c) (DC:78) [09:03:17:779]: PROPERTY CHANGE: Adding PackageCode property. Its value is '{AF82ECE5-8139-4C1F-B2E3-78D336CA5E3B}'.
MSI (c) (DC:78) [09:03:17:779]: Product Code passed to Engine.Initialize:
MSI (c) (DC:78) [09:03:17:779]: Product Code from property table before transforms: '{E713FEDC-4C78-4BDE-8E91-77D9B7C23C50}'
MSI (c) (DC:78) [09:03:17:779]: Product Code from property table after transforms: '{E713FEDC-4C78-4BDE-8E91-77D9B7C23C50}'
MSI (c) (DC:78) [09:03:17:779]: Product not registered: beginning first-time install
MSI (c) (DC:78) [09:03:17:780]: PROPERTY CHANGE: Adding ProductState property. Its value is '-1'.
MSI (c) (DC:78) [09:03:17:780]: Entering CMsiConfigurationManager::SetLastUsedSource.
MSI (c) (DC:78) [09:03:17:780]: User policy value 'SearchOrder' is 'nmu'
MSI (c) (DC:78) [09:03:17:789]: Adding new sources is allowed.
MSI (c) (DC:78) [09:03:17:789]: PROPERTY CHANGE: Adding PackagecodeChanging property. Its value is '1'.
MSI (c) (DC:78) [09:03:17:789]: Package name extracted from package path: 'evm20.msi'
MSI (c) (DC:78) [09:03:17:789]: Package to be registered: 'evm20.msi'
MSI (c) (DC:78) [09:03:17:802]: Note: 1: 2262 2: AdminProperties 3: -2147287038
MSI (c) (DC:78) [09:03:17:802]: Machine policy value 'DisableMsi' is 0
MSI (c) (DC:78) [09:03:17:802]: Machine policy value 'AlwaysInstallElevated' is 0
MSI (c) (DC:78) [09:03:17:802]: User policy value 'AlwaysInstallElevated' is 0
MSI (c) (DC:78) [09:03:17:802]: Product installation will be elevated because user is admin and product is being installed per-machine.
MSI (c) (DC:78) [09:03:17:802]: Running product '{E713FEDC-4C78-4BDE-8E91-77D9B7C23C50}' with elevated privileges: Product is assigned.
MSI (c) (DC:78) [09:03:17:802]: PROPERTY CHANGE: Adding REBOOT property. Its value is 'ReallySuppress'.
MSI (c) (DC:78) [09:03:17:802]: PROPERTY CHANGE: Adding REBOOTPROMPT property. Its value is 'Suppress'.
MSI (c) (DC:78) [09:03:17:802]: PROPERTY CHANGE: Adding WISE_SETUP_EXE_PATH property. Its value is 'C:\Users\demo\AppData\Local\INtime_Installer\evm20.exe'.
MSI (c) (DC:78) [09:03:17:802]: PROPERTY CHANGE: Adding CURRENTDIRECTORY property. Its value is 'C:\Users\demo\AppData\Local\INtime_Installer'.
MSI (c) (DC:78) [09:03:17:802]: PROPERTY CHANGE: Adding CLIENTUILEVEL property. Its value is '0'.
MSI (c) (DC:78) [09:03:17:802]: PROPERTY CHANGE: Adding CLIENTPROCESSID property. Its value is '3548'.
MSI (c) (DC:78) [09:03:17:802]: TRANSFORMS property is now:
MSI (c) (DC:78) [09:03:17:802]: PROPERTY CHANGE: Adding VersionDatabase property. Its value is '400'.
MSI (c) (DC:78) [09:03:17:802]: SHELL32::SHGetFolderPath returned: C:\Users\demo\AppData\Roaming
MSI (c) (DC:78) [09:03:17:802]: SHELL32::SHGetFolderPath returned: C:\Users\demo\Favorites
MSI (c) (DC:78) [09:03:17:803]: SHELL32::SHGetFolderPath returned: C:\Users\demo\AppData\Roaming\Microsoft\Windows\Network Shortcuts
MSI (c) (DC:78) [09:03:17:803]: SHELL32::SHGetFolderPath returned: C:\Users\demo\Documents
MSI (c) (DC:78) [09:03:17:803]: SHELL32::SHGetFolderPath returned: C:\Users\demo\AppData\Roaming\Microsoft\Windows\Printer Shortcuts
MSI (c) (DC:78) [09:03:17:803]: SHELL32::SHGetFolderPath returned: C:\Users\demo\AppData\Roaming\Microsoft\Windows\Recent
MSI (c) (DC:78) [09:03:17:803]: SHELL32::SHGetFolderPath returned: C:\Users\demo\AppData\Roaming\Microsoft\Windows\SendTo
MSI (c) (DC:78) [09:03:17:803]: SHELL32::SHGetFolderPath returned: C:\Users\demo\AppData\Roaming\Microsoft\Windows\Templates
MSI (c) (DC:78) [09:03:17:803]: SHELL32::SHGetFolderPath returned: C:\ProgramData
MSI (c) (DC:78) [09:03:17:803]: SHELL32::SHGetFolderPath returned: C:\Users\demo\AppData\Local

Figure 4.12 - eVM Installation Log

13. After you click **Finish** in the dialog shown in Figure 4.10, the Wizard will prompt you that you now need to restart your system, see Figure 4.13. A restart is required, so eVM can take the memory that it needs for eVM and the guest OS from Windows. In general, resizing Windows available memory requires a restart of the Windows OS. Click **Yes**, and wait for the system to reboot.

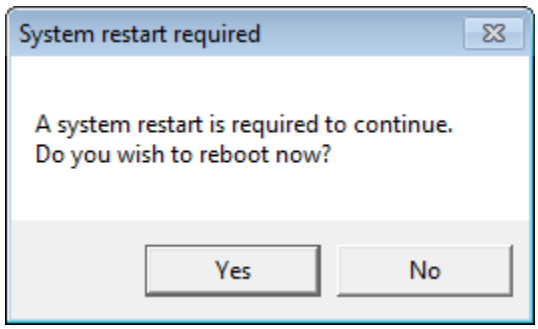


Figure 4.13 - Restart Prompt

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

14. After the system has rebooted, wait for the eVM services to start, which will be indicated by the 'e' icon showing up in the system tray, see Figure 4.14.

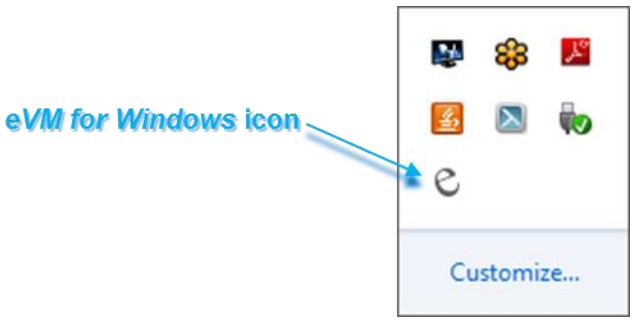


Figure 4.14 - eVM System Tray Icon

15. Click on the **eVM tray icon** and select **eVM Control Panel** to launch the eVM Control Panel, see Figure 4.15.

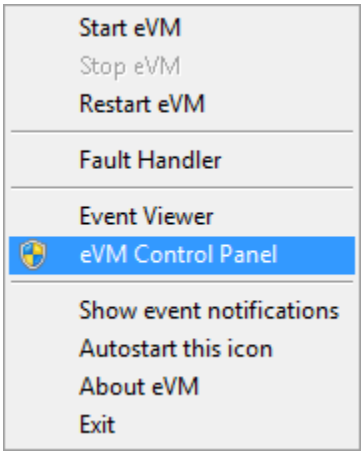


Figure 4.15 Launching the eVM Control Panel from the system tray

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

16. Launching the Control Panel from the system tray will bring up the eVM Configuration Panel with 4 icons enabling you setup and run eVM for Windows.

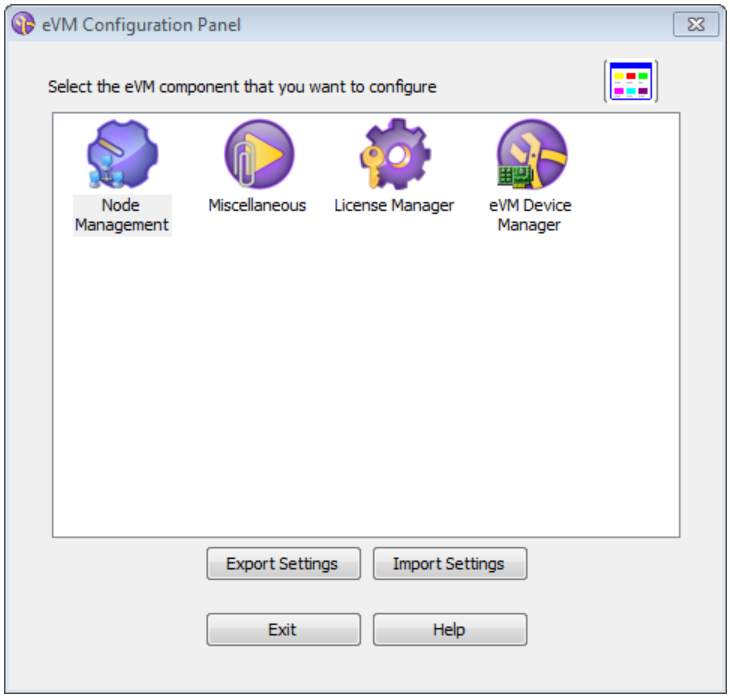


Figure 4.16 eVM Configuration Panel

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

17. Clicking on the *Node Management* icon in the *eVM Configuration Panel* brings up the *Guest* configuration tab of the *Node Management* screen., if you chose the *Complete eVM Installation*, you will see the system iRMX listed in the Guest selection line and the I/O setup below it for the selected guest. Guest starts running when the *Start Guest* button in clicked. Clicking on the *System* tab will display eVM Configuration setup and the last tab, the *Log* tab will display some of the install log.

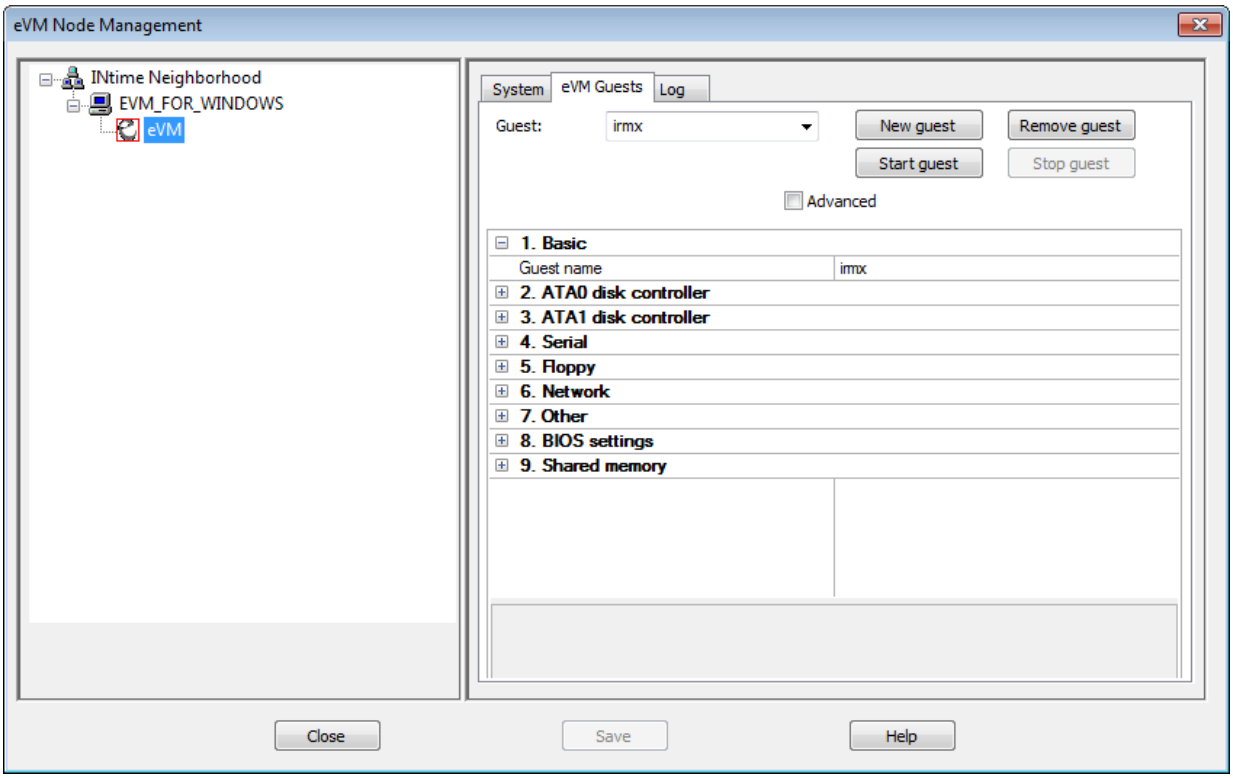


Figure 4.17 - eVM Control Panel with iRMX Guest RTOS Installed

With the *Complete Installation*, eVM been installed with the 2 virtual serial ports and sample guest RTOS, iRMX, complete with a sample configuration.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

18. Going back to the *eVM Configuration Panel* and clicking on the *eVM Device Manager* will list out the Windows devices that can be passed to the guest Operating system (OS) that is running on eVM (Fig 4.19) and any device that has already been passed to the guest OS. When you select *All devices* under the *View* menu tab, all the devices including those that cannot be transferred to eVM are listed (Fig 4.20). Note that this screen includes the TenAsys virtual Ethernet Port and virtual COM ports. Left hand-side of the window, lists out installed devices and the legacy (hardware) interrupt they use.

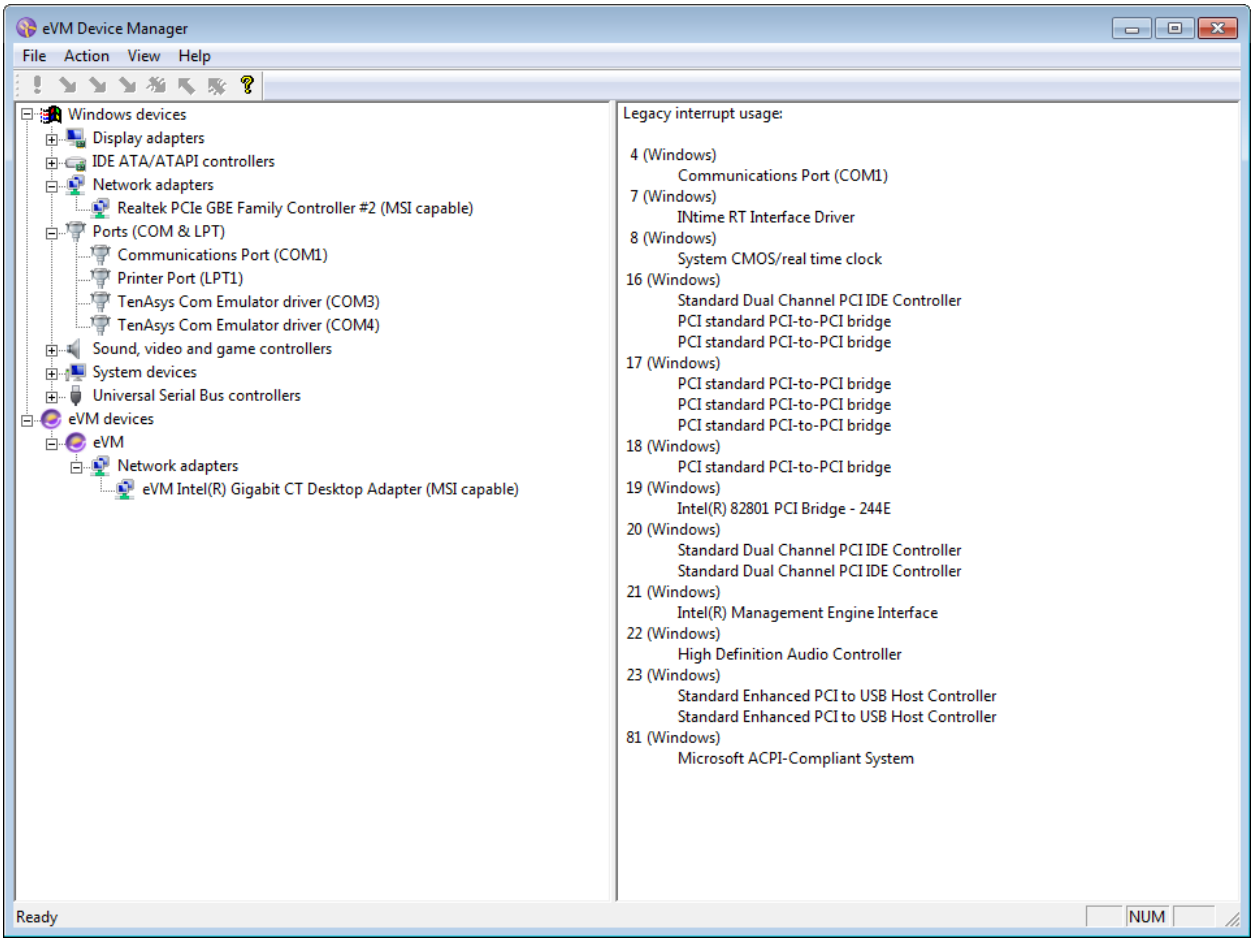


Figure 4.18 Device Manager showing the Windows devices that can be passed to eVM

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

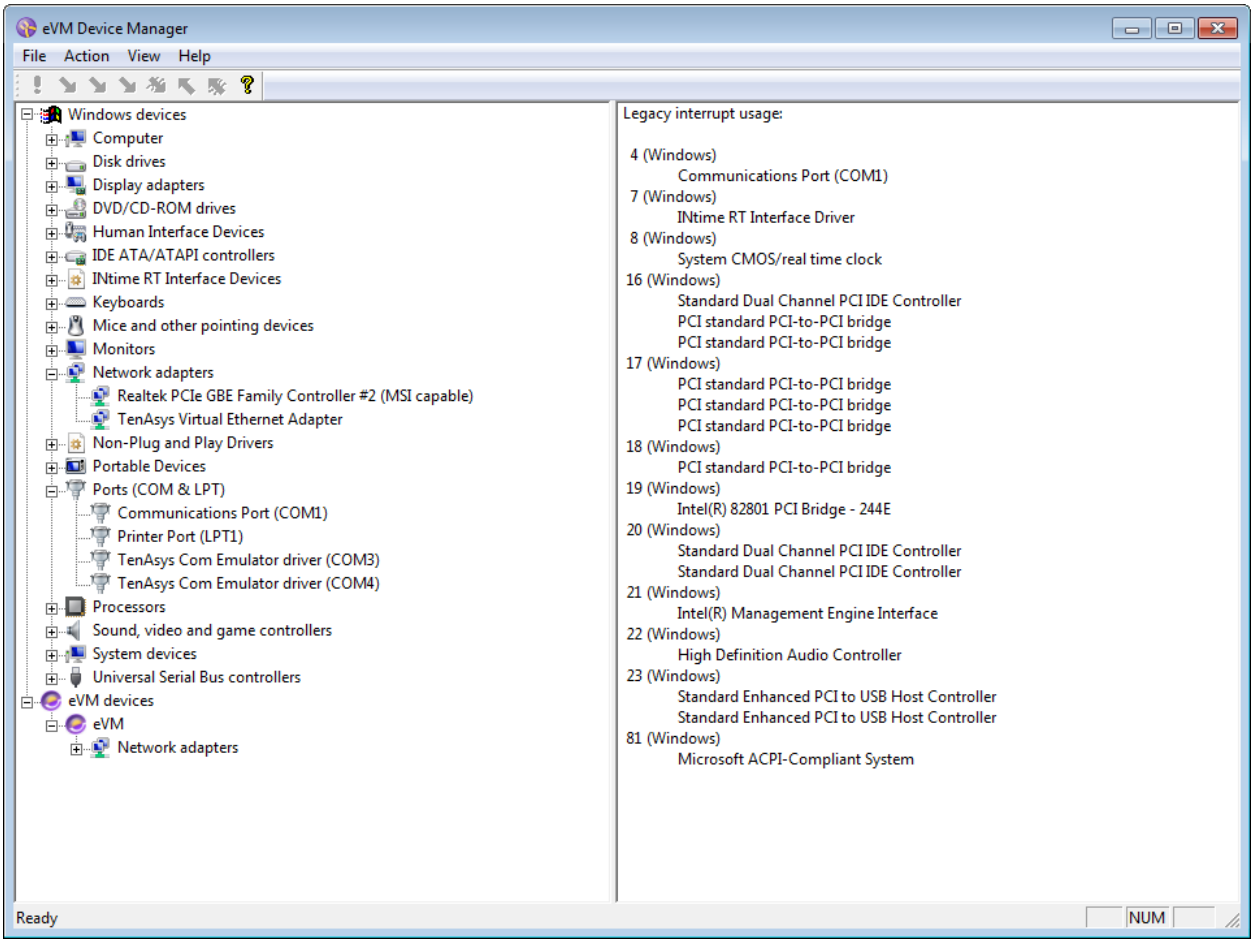


Figure 4.19 Device Manager showing all the devices, including ones that cannot be passed to eVM (the guest OS)

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

19. You can compare the eVM's Device Manager to that display by opening *Windows Explorer*, right-clicking on *Manage* and the clicking on *Device Manager*. Figure 4.21. You should see the same list of devices allocated to Windows. Devices that have been allocated to eVM will still appear in the Windows Device Manager but with the prefix "eVM".

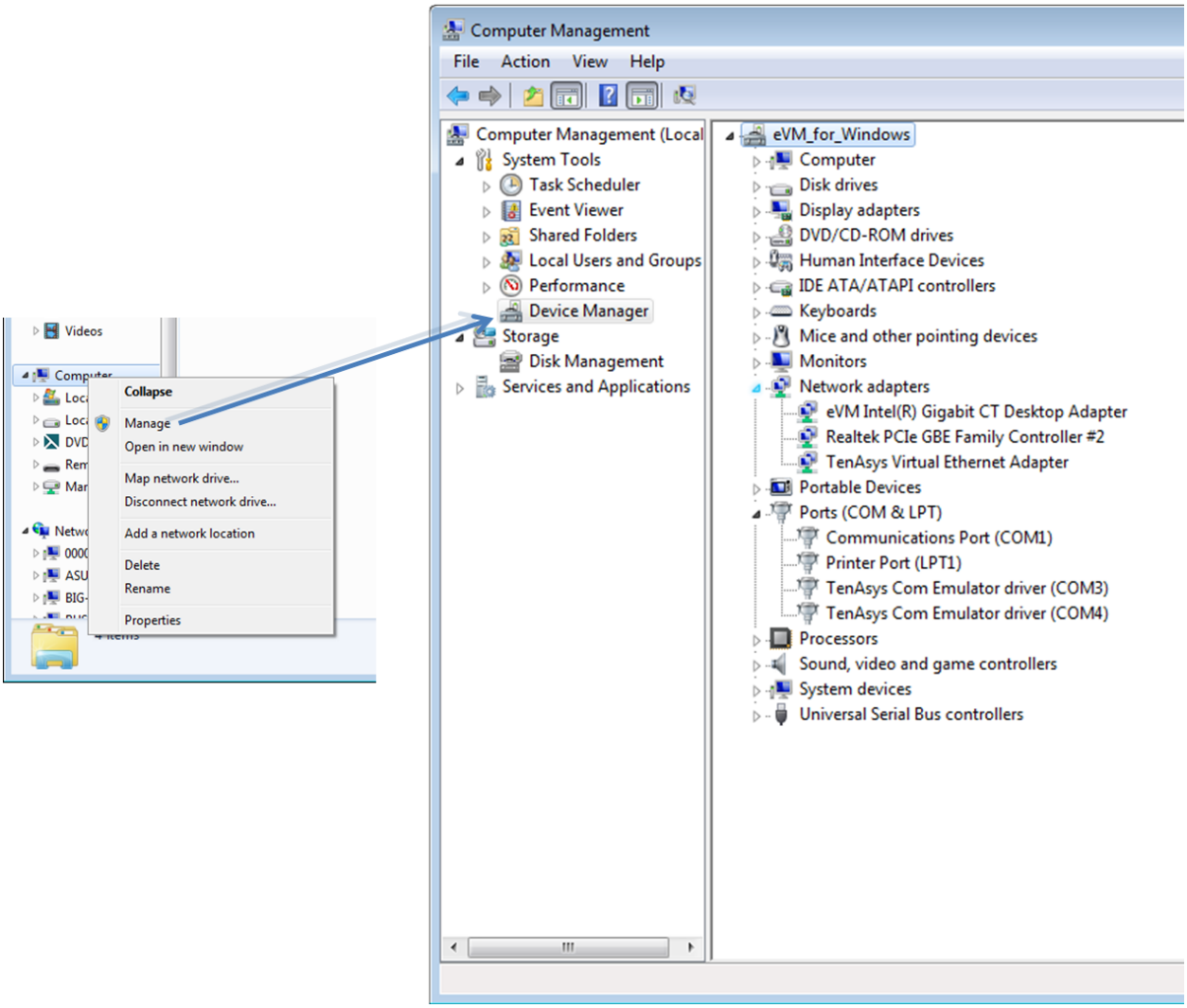


Figure 4.20 Windows Device Manager

You can also see that the TenAsys Virtual Ethernet Adapter has been installed, as well as, the two virtual serial ports. The property of each virtual serial port was to use the next available COM device, so you can see that in this system configuration, it resulted in the virtual serial ports becoming COM3 and COM4. We will discuss the configuration of the virtual Ethernet Adapter later.

4.2 Virtual Ports

Using eVM allows two discrete hardware platforms, each with its own operating system and applications, to be consolidated onto a single hardware platform. Not only does this provide a

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

reduction in hardware required to host the two systems, it also provides a unique opportunity to simplify communications and connectivity between them. Instead of having real hardware communications devices on each discrete hardware platform that are hard-wired together, eVM can provide a virtual software connection between the two systems, further reducing the hardware requirement to host the two interconnected systems.

4.2.1 Installation of Virtual Serial Ports

When eVM was installed, 2 virtual serial ports were also installed. If you want to add additional serial ports, eVM supports up to 4 virtual serial ports, you can add them through the *eVM Configuration Panel*, then selecting *Node Management* icon and selecting the *System* tab. Let's look at a system with two (2) serial ports installed and look at how to install and configure another virtual serial port, a virtual COM device (you may have to scroll down until you see the Hardware category). Each virtual serial port provides a virtual COM device for the virtual machine that the guest OS can use for serial communications and a corresponding virtual COM device in Windows that will looks like a standard COM device that is wired to the virtual COM device in the virtual machine. eVM manages this connection between the two COM devices using a shared memory interface.

1. Open the eVM Control Panel, and in the left panel under System, click **Configuration,** Figure 4.21

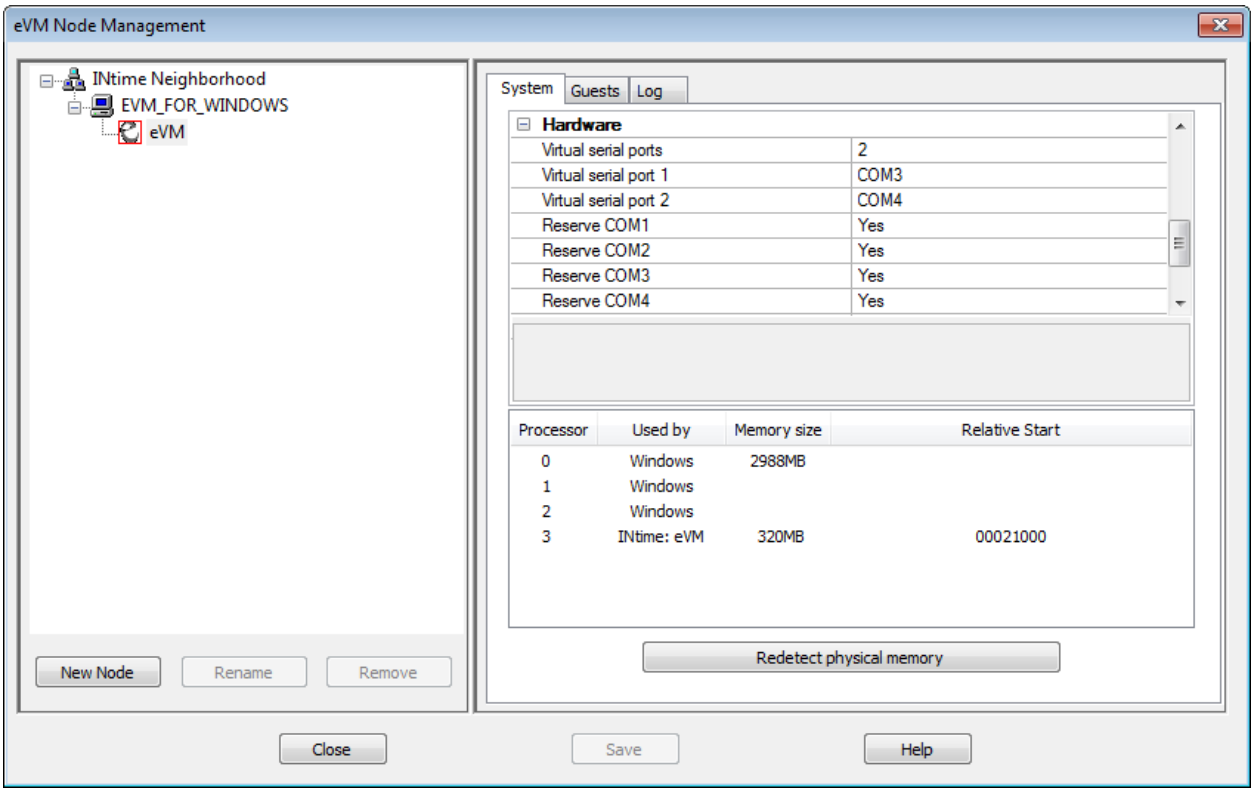


Figure 4.21 eVM Control Panel Showing System Configuration with No Virtual Serial Ports

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

In the right pane of the eVM control Panel, the system configuration parameters are shown. ***Note that the Virtual serial ports parameter is 2 for this example.***

2. Double-click on the '**2**' value in the box next to *Virtual serial ports*, and enter '**3**' Figure 4.22

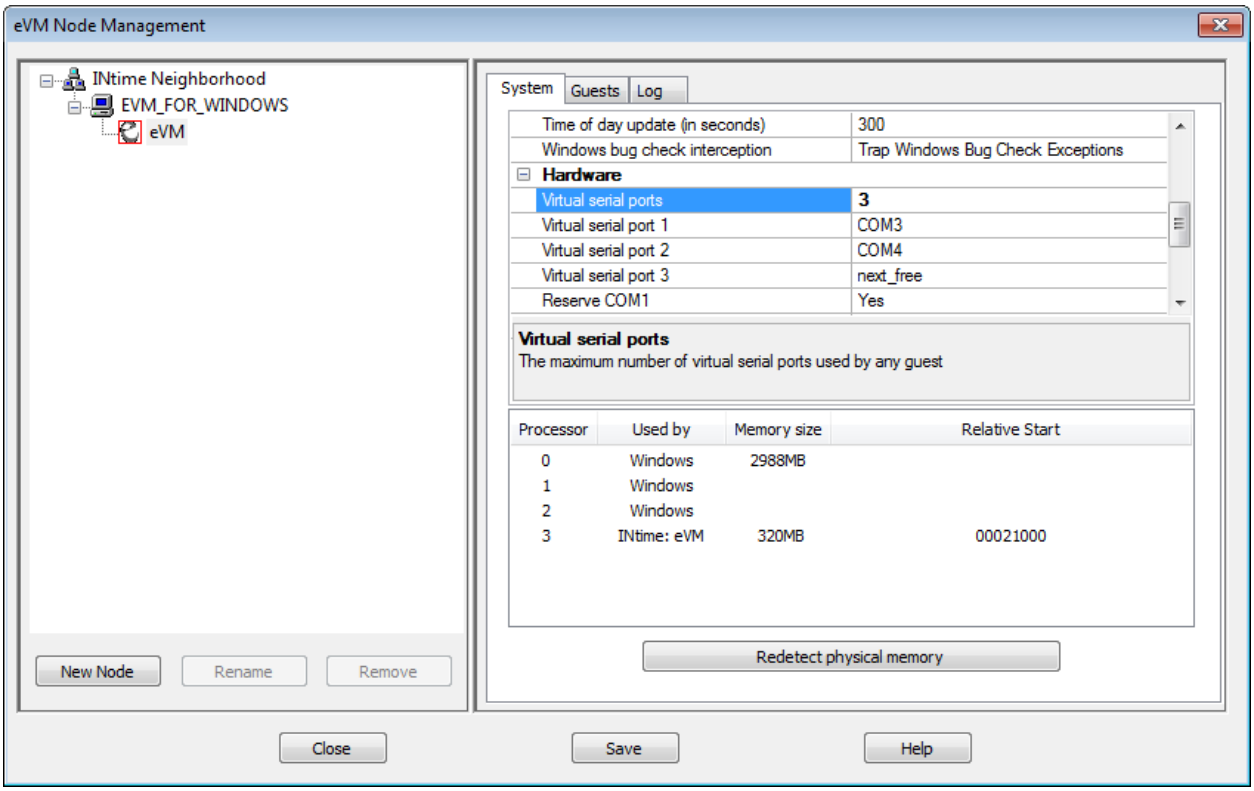


Figure 4.22 - eVM Control Panel, Virtual Serial Port Added

You will note that a new parameter has been added under *Virtual serial ports*, *Virtual serial port 3*. This is the new virtual COM device that has been added, and the default is to let eVM assign the COM port number to the next available. If a specific COM port number would be desired, it could be specified with the *Virtual serial port 3* parameter. If you click on the dropdown on the box next to Virtual serial port 1, you will see the COM port choices, Figure 4.23. We will leave the selection for *next_free* for this example.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

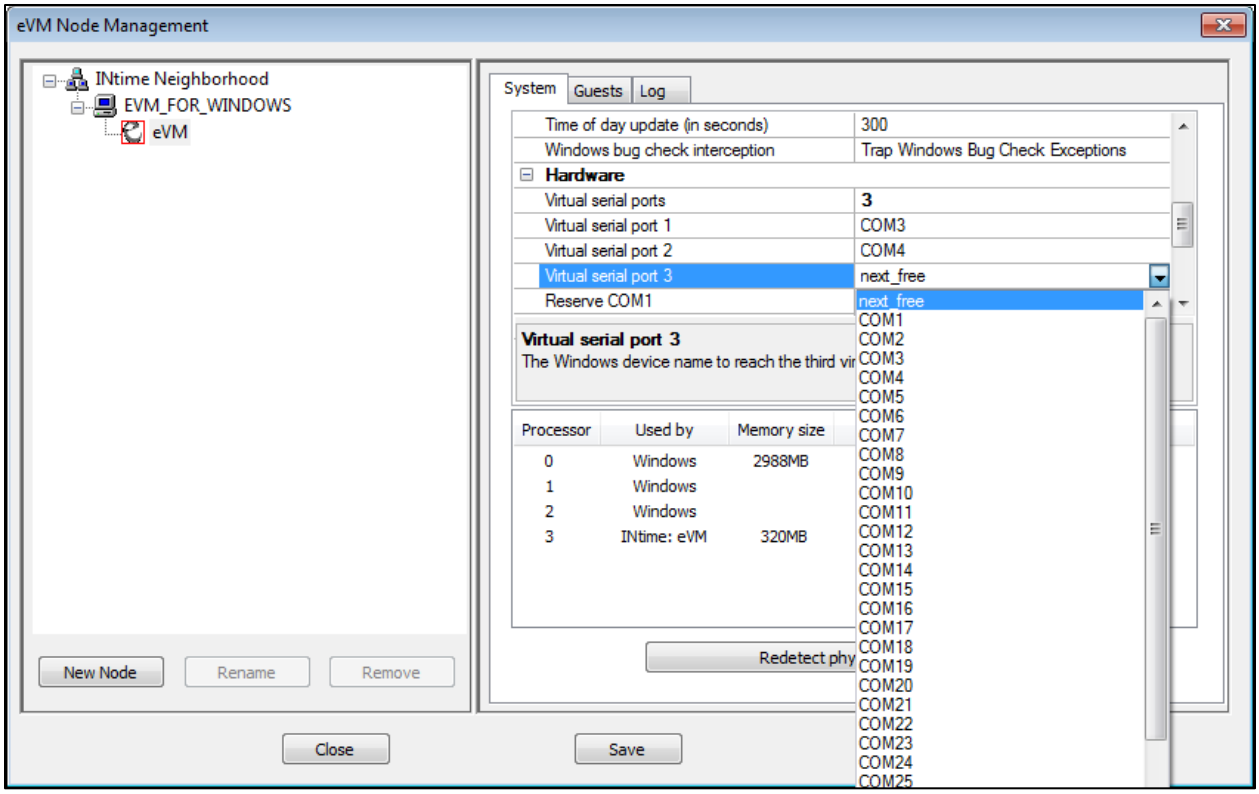


Figure 4.23 Virtual Serial Port COM Port Selections

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

3. To apply this configuration change, you must save the new configuration. From the eVM Control Panel dropdown, select **Save** button at the bottom of the window**.** See Figure 4.24. Note: You might get a screen prompt as shown in Figure Figure 4.24

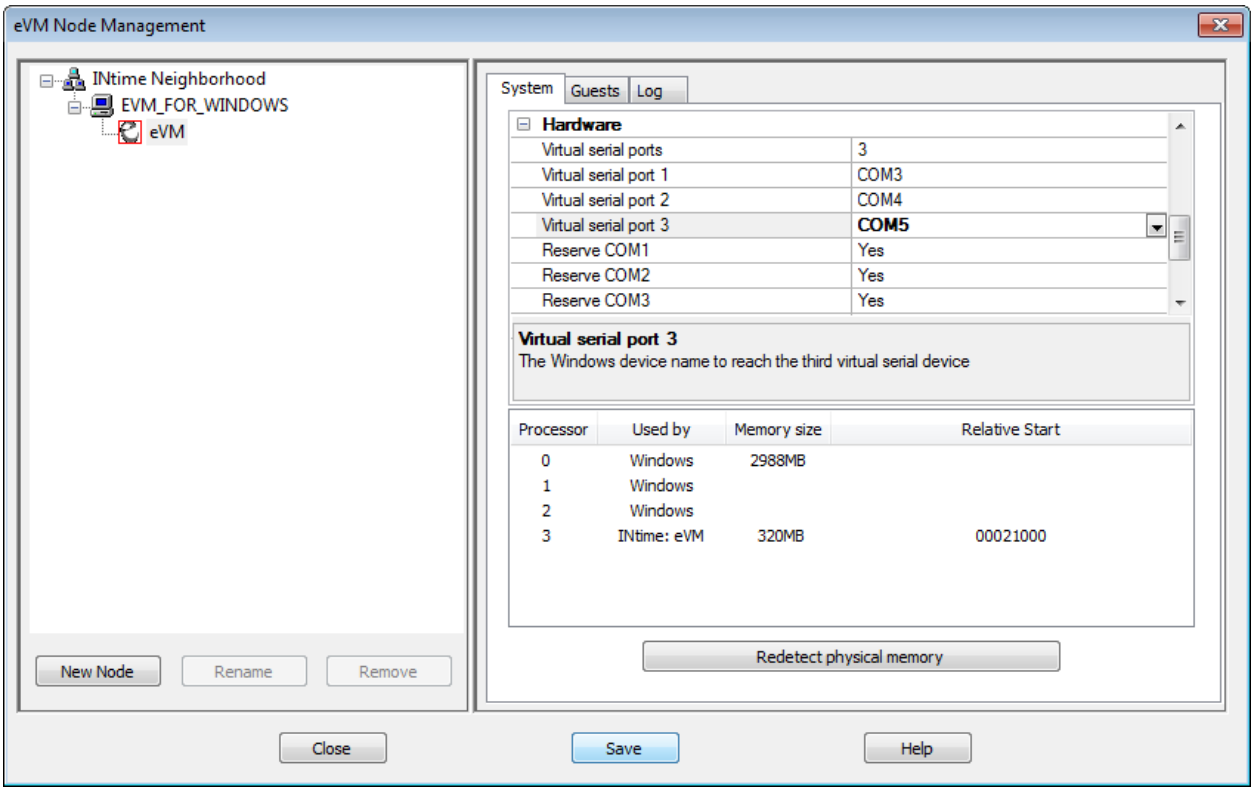


Figure 4.24 Saving eVM Control Panel Configuration

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

4. Now, let's confirm that the virtual serial port has been added. Open *Windows Explorer*, right-click on *Manage* and then click on *Device Manager*. You will now see three (3) virtual serial port: *TenAsys Com Emulator driver* with the 3rd associated with to *COM5.* See Figure 4.25.

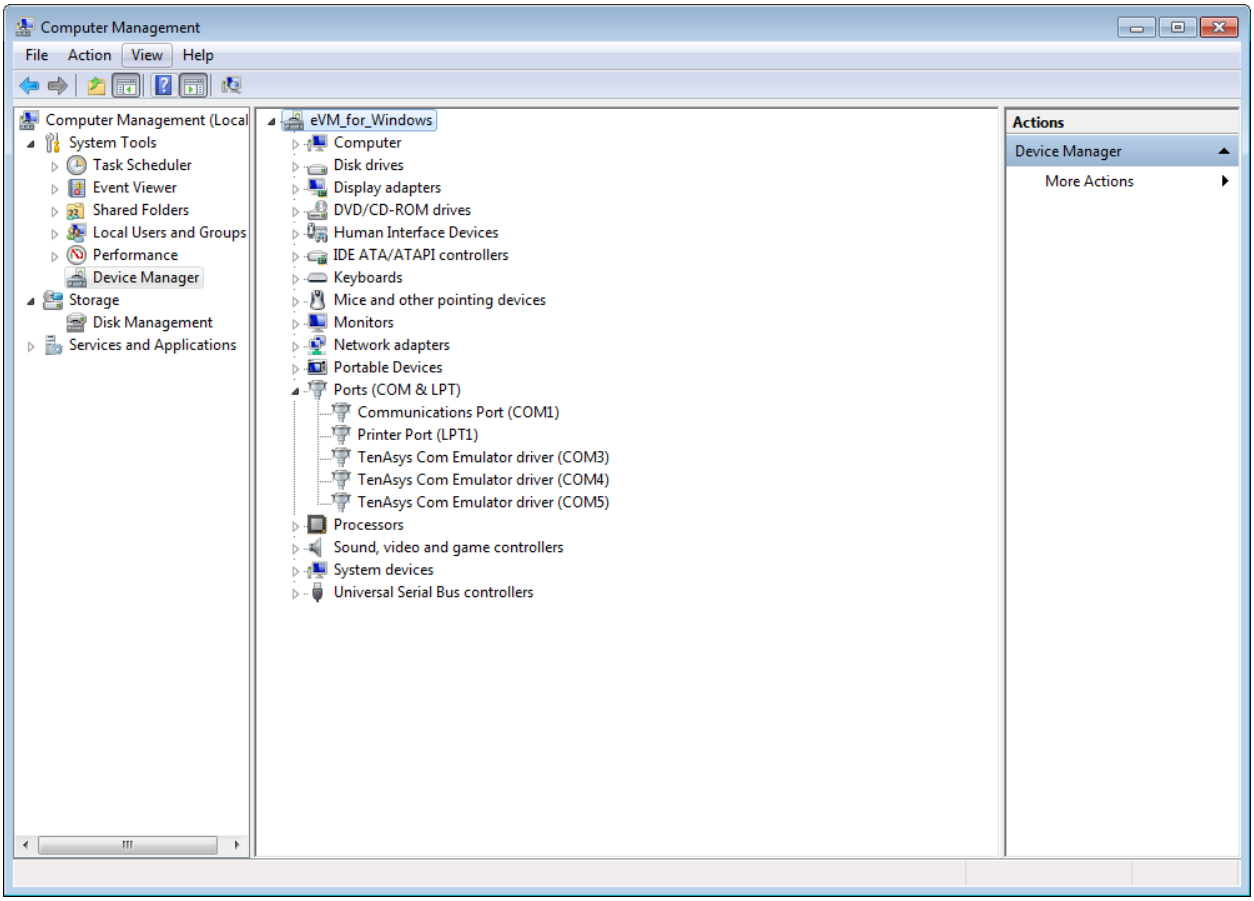


Figure 4.25 - Windows Device Manager Showing Virtual Serial Port

Applications in Windows can now access this virtual COM port by connecting to COM5.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

5. Now, let's connect the virtual COM ports in the guest OS to the virtual COM ports of the Windows. Open eVM Control Panel, select the guest OS in the left pane (in this example it is iRMX), and expand the serial section in the right panel, see Figure 4.26.

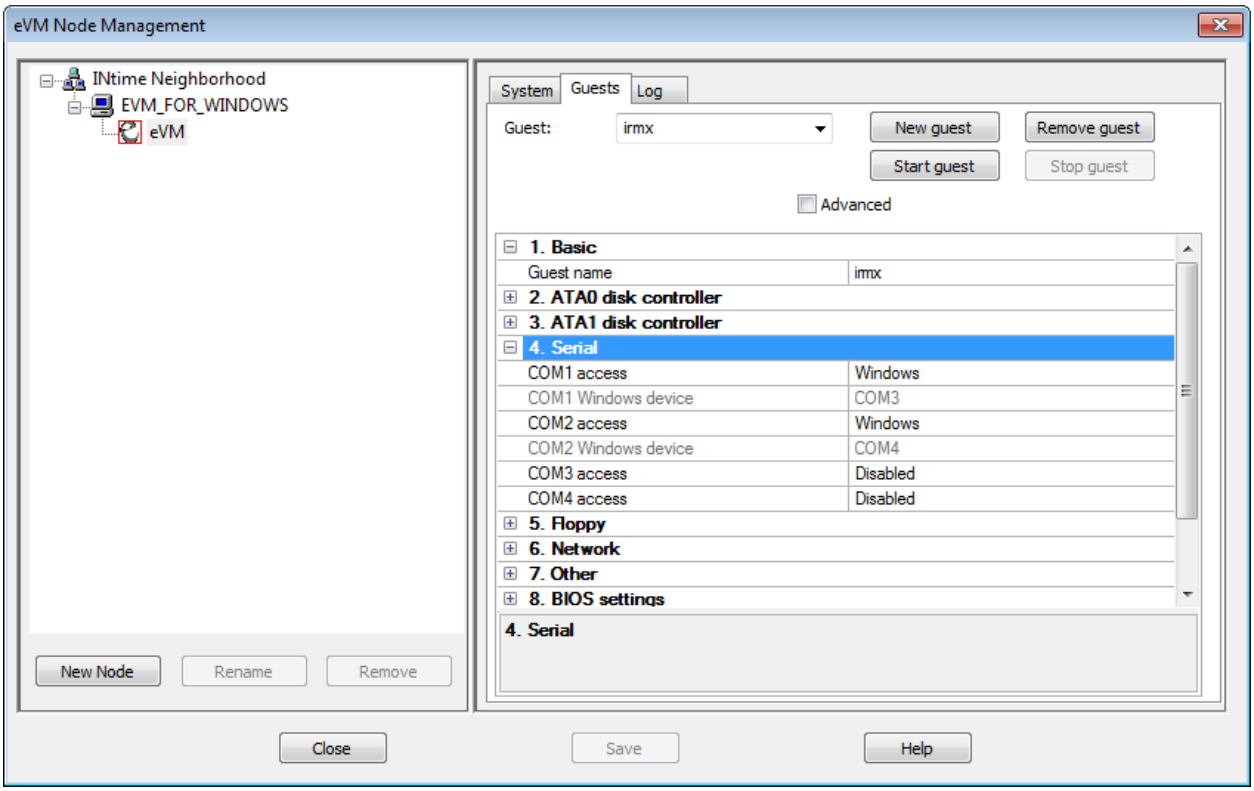


Figure 4.26 - eVM Control Panel Showing Guest Serial Device Settings

6. For each COM port that is to be connected to a virtual COM device in Windows, select the **COMn access**, and select **Windows**. In this example, iRMX has been set up for COM1 and COM1 to access Windows virtual COM devices.

7. Then to select the COMn device (where n = 1, 2, 3, ..) in Windows that the guest COMm device will connect to, select **COMn Windows device**, and select the desired COMn device for Windows. For this example iRMX COM1 will be connected to Windows COM3, and iRMX COM2 will be connected to Windows COM4.

4.2.2 Configuring the Virtual Ethernet Device

In this example we look at the case where we have two platforms interconnected via Ethernet that are now consolidated onto one platform, with the Ethernet connection being managed by eVM via virtual Ethernet devices. The consolidated system will consist of Windows as the host OS and Windows CE as the guest RTOS.

There are two parts to setting up eVM to manage the Virtual Ethernet connection, the virtual Ethernet device on the Windows side, and the virtual Ethernet device on the guest OS side.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

Configuring the Windows side:

The follow step-by-step guide is for Windows 8/7/Vista.

1. Open the **Windows Control Panel** and click on **Network and Sharing Center**, Figure 4.27.

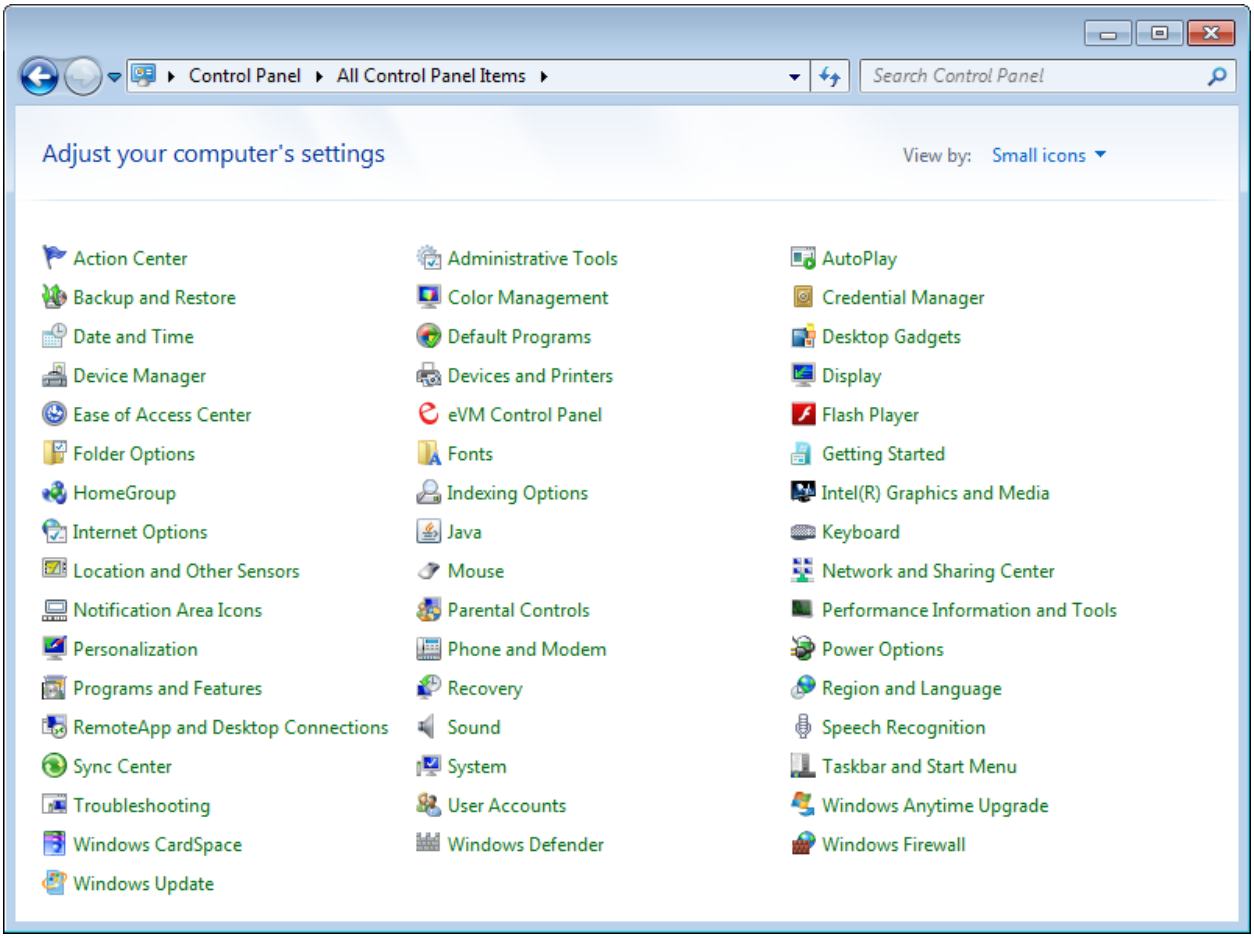


Figure 4.27 - Windows Control Panel, Select Network

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

2. In the upper left of the *Network and Sharing Center*, click on **Change adapter settings,** Figure 4.28 .

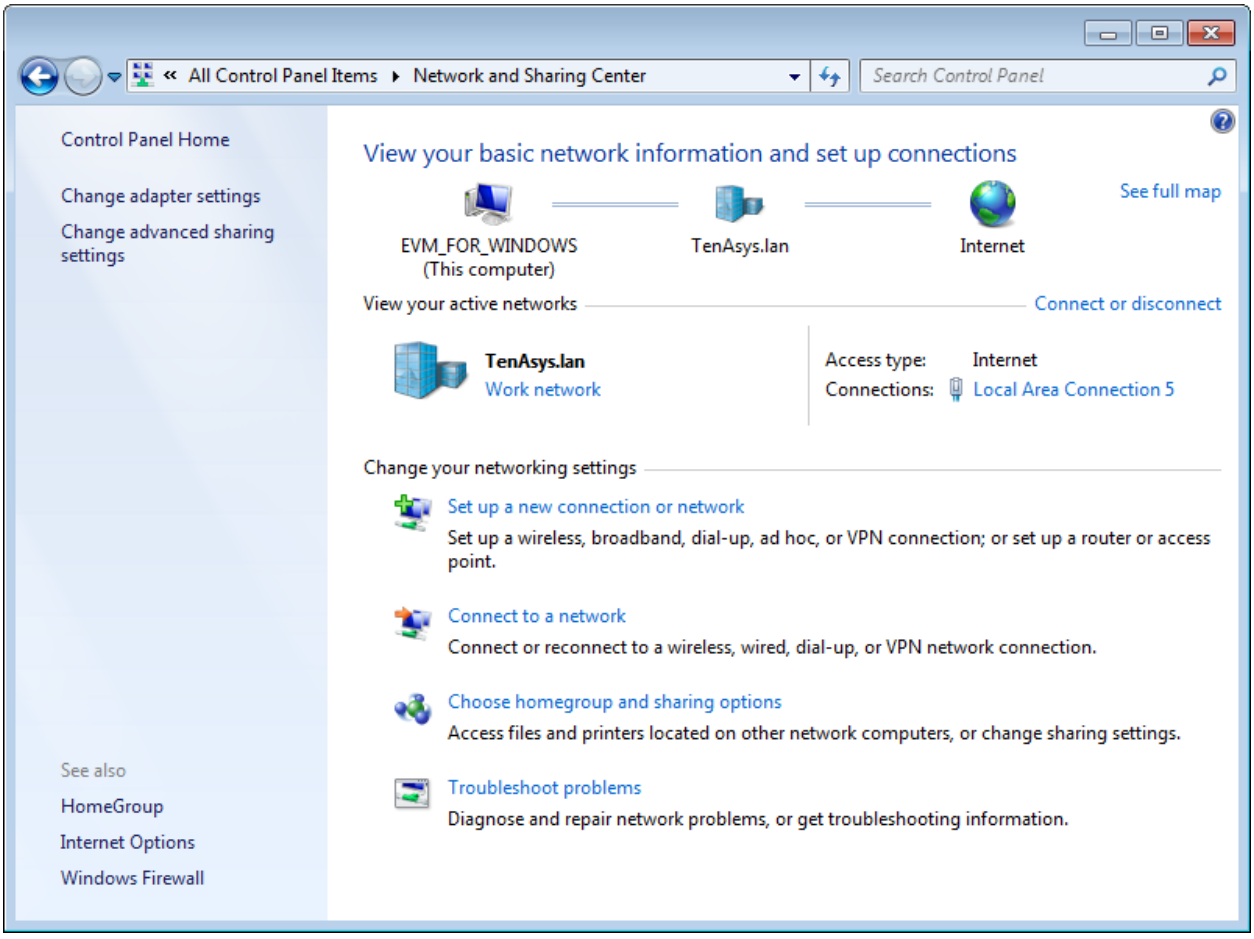

« All Control Panel Items ▸ Network and Sharing Center
Search Control Panel
Control Panel Home
Change adapter settings
Change advanced sharing settings
View your basic network information and set up connections
See full map
EVM_FOR_WINDOWS
(This computer)
TenAsys.lan
Internet
View your active networks
Connect or disconnect
TenAsys.lan
Work network
Access type: Internet
Connections: Local Area Connection 5
Change your networking settings
Set up a new connection or network
Set up a wireless, broadband, dial-up, ad hoc, or VPN connection; or set up a router or access point.
Connect to a network
Connect or reconnect to a wireless, wired, dial-up, or VPN network connection.
Choose homegroup and sharing options
Access files and printers located on other network computers, or change sharing settings.
Troubleshoot problems
Diagnose and repair network problems, or get troubleshooting information.
See also
HomeGroup
Internet Options
Windows Firewall

Figure 4.28 - Windows Network and Sharing Center

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

3. Now you will see *Network Connections*, and one of the network connections displayed is the *TenAsys Virtual Ethernet Adapter,* Figure 4.29. Double-click on the **TenAsys Virtual Ethernet Adapter**.

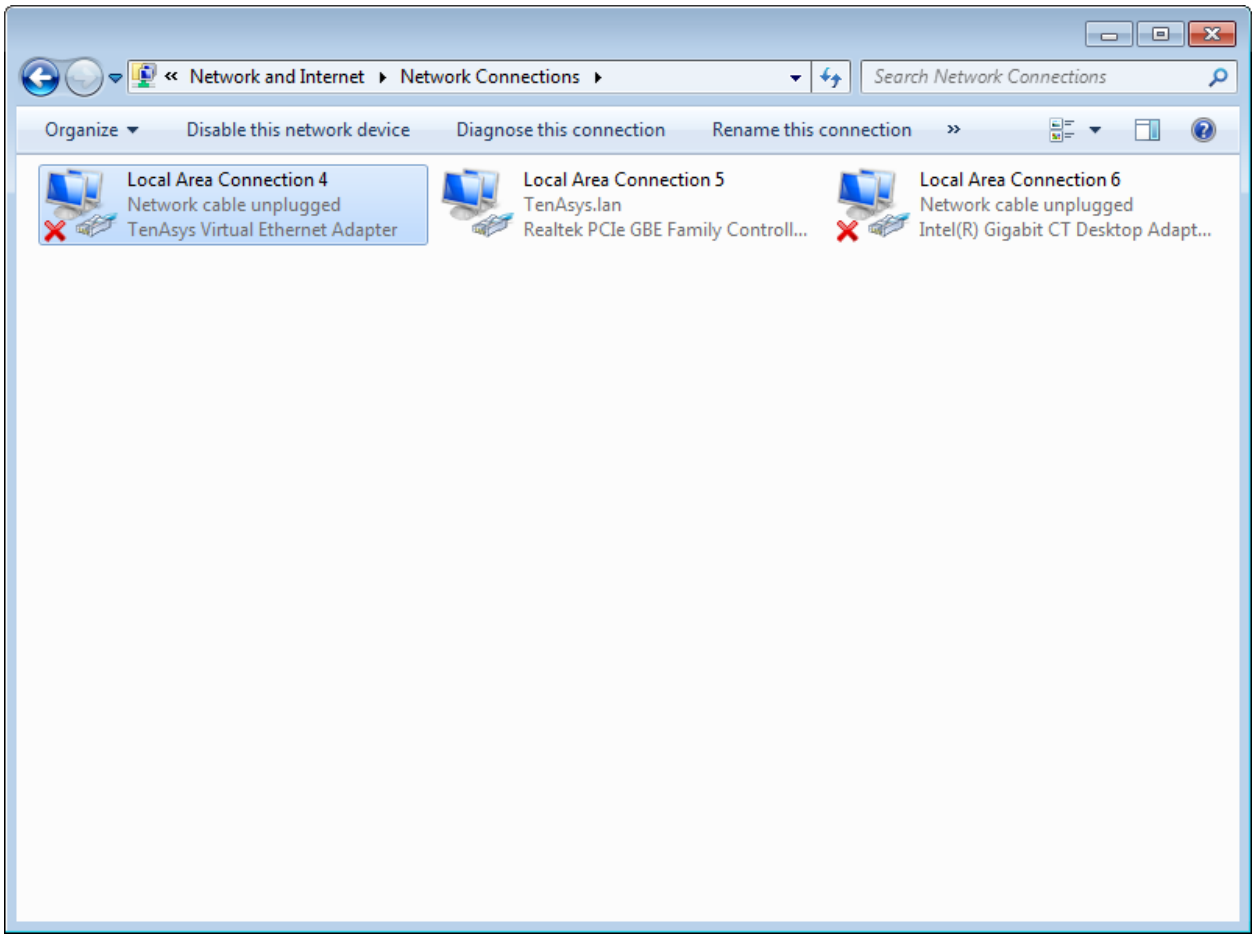


Figure 4.29 - Windows Network Connections

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

4. You will now see the *Local Area Connection Properties* for the *TenAsys virtual Ethernet Adapter*. We will be using TCP/IPv4, so uncheck the checkbox for **Internet Protocol 6 (TCP/IPv6)**, Figure 4.30.

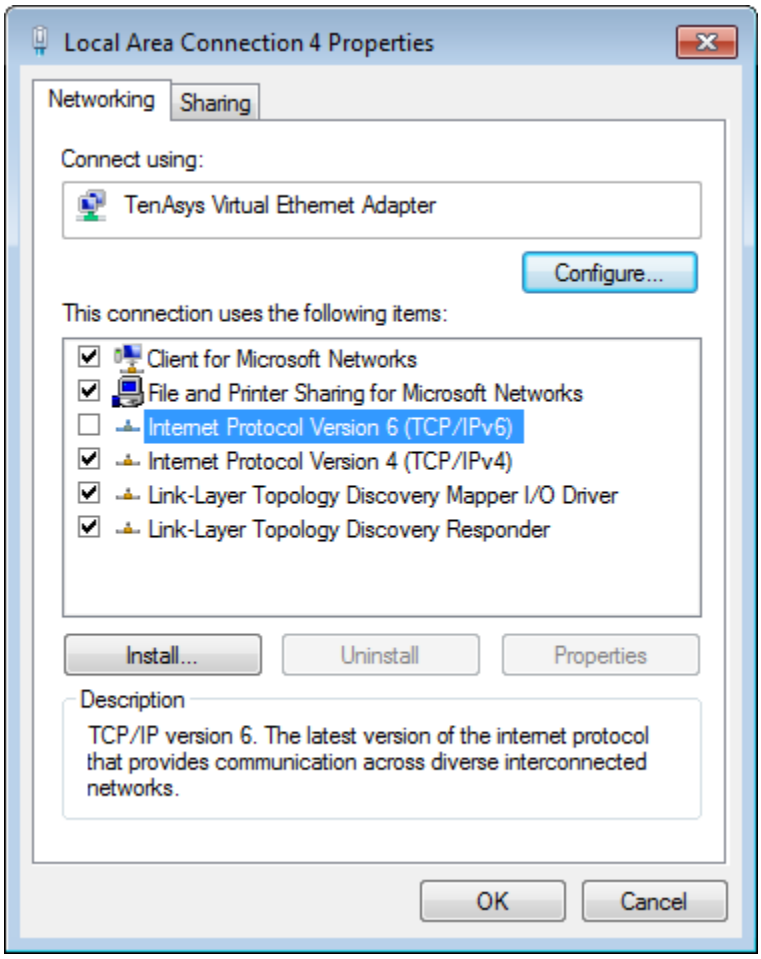


Figure 4.30 - TenAsys Virtual Ethernet Adapter, Windows Local Area Connection Properties

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

5. Select Internet Protocol Version 4(TCP/IPv4), and click the Properties button, Figure 4.31.

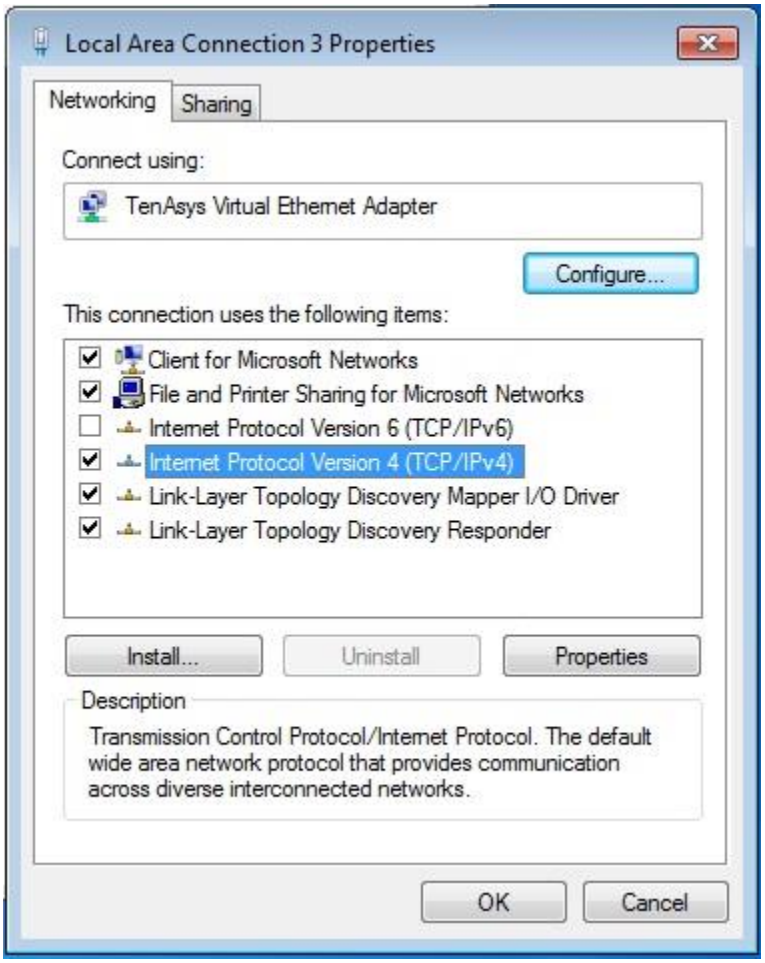


Figure 4.31 - Selecting TenAsys Virtual Ethernet Adapter TCP/IPv4 Properties

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

6. In the *Internet Protocol Version 4 (TCP/IPv4) Properties* window, select the **Use the following IP address:** radio button, and enter **10.0.0.1** in the IP address. Keep the default Subnet mask: of **255.0.0.0**,Figure 4.32.

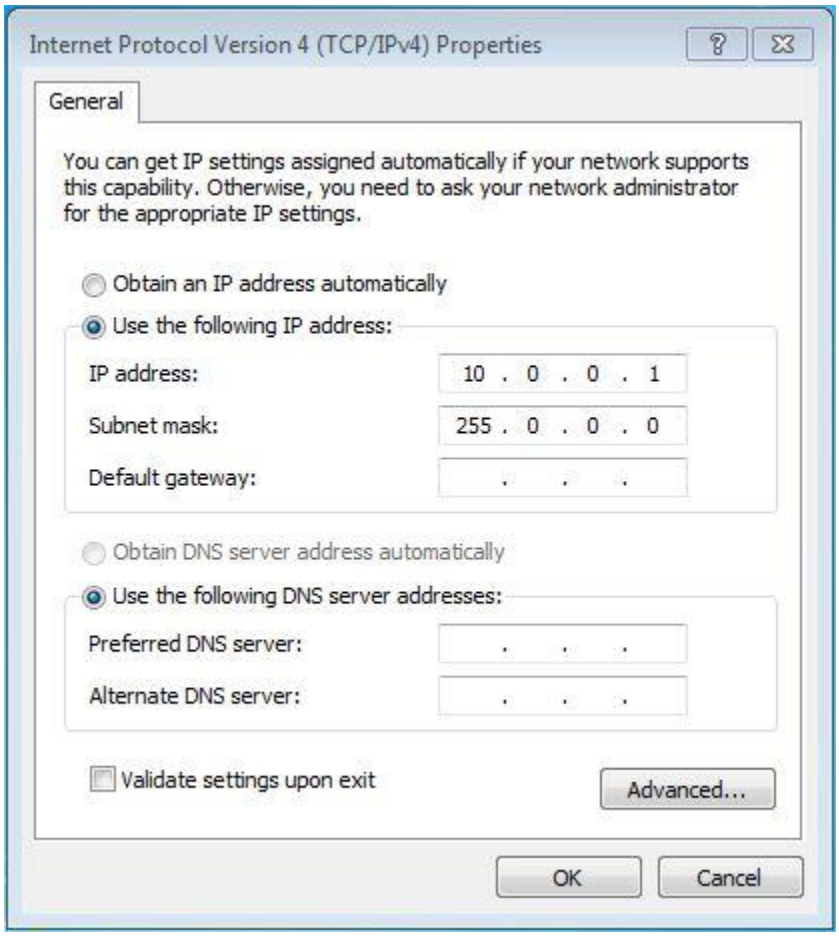

Internet Protocol Version 4 (TCP/IPv4) Properties
General
You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.
Obtain an IP address automatically
Use the following IP address:
IP address:
10 . 0 . 0 . 1
Subnet mask:
255 . 0 . 0 . 0
Default gateway:
Obtain DNS server address automatically
Use the following DNS server addresses:
Preferred DNS server:
Alternate DNS server:
Validate settings upon exit
Advanced...
OK
Cancel

Figure 4.32 - Setting the Static IP Address for the TenAsys Virtual Ethernet Adapter

7. To accept this new IP address setting, click **OK**.

8. Close the *Network Connections* window, and the Windows side of the virtual Ethernet connection configuration is complete.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

Configuring the guest OS side:

1. Open **eVM Control Panel**, select the **guest OS**, and select the **Network** section, which has an item named *Virtual Ethernet (NE2000)*,Figure 4.33.

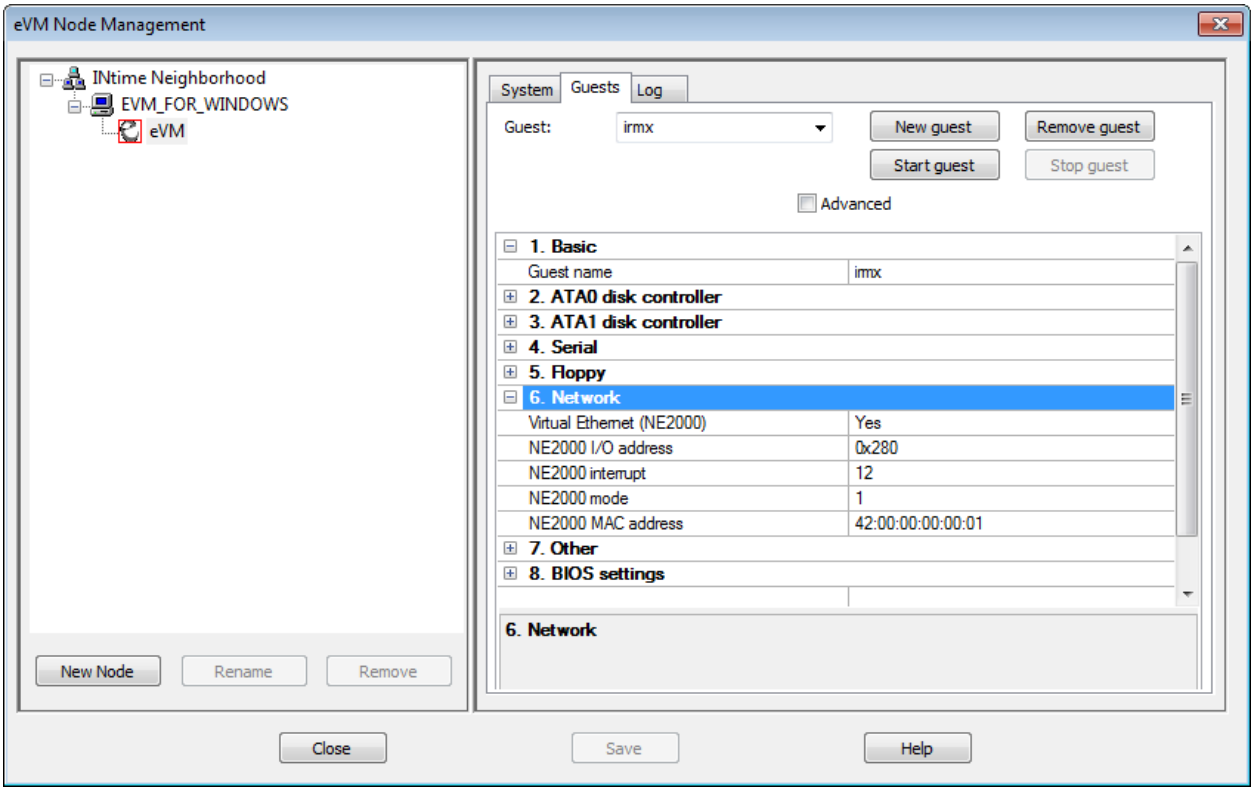


Figure 4.33 - eVM Control Panel, Guest OS Virtual Ethernet Device Configuration

1. In the box next to *Virtual Ethernet (NE2000)*, select **Enabled** to enable the virtual Ethernet device.

2. You will now see the configuration for the virtual Ethernet adapter for the virtual machine. Match the I/O address, and interrupt to what the Guest OS expects.

Note: we will discuss more about the guest OS device parameters in general, and with respect to QNX and Windows CE specifically in Section 4.5.

3. The guest OS needs to be configured to use an IP address on the same sub-net as the virtual Ethernet adapter on the Windows side. In this example with iRMX , we entered 10.0.0.2 as the IP address in the boot loader. The guest OS now sees a virtual NE2000 Ethernet device at I/O 0x280, interrupt 12 and configures its NE2000 driver to use IP address 10.0.0.2 for that device.

The actual guest OS needs to talk to the IP address of the Windows side (10.0.0.1); and when the Windows side responds, it needs to know to talk to the IP address of the guest OS side

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

(10.0.0.2). When we did our Windows CE example, we told the Windows host to find a client at 10.0.0.2 and send the image to it.

4.3 Running the Sample Guest RTOS, iRMX

Now that we have eVM installed and a virtual COM device installed and configured, let's run the sample guest RTOS, iRMX.

1. In Windows, open a terminal emulator program, and configure it to connect to the first virtual COM device. Recall from Figure 4.26 that the first virtual COM device was COM3 in our example.

2. Open **eVM Control Panel** and select **iRMX** under eVM guests, Figure 4.34.

3. Click the **Start Guest** button

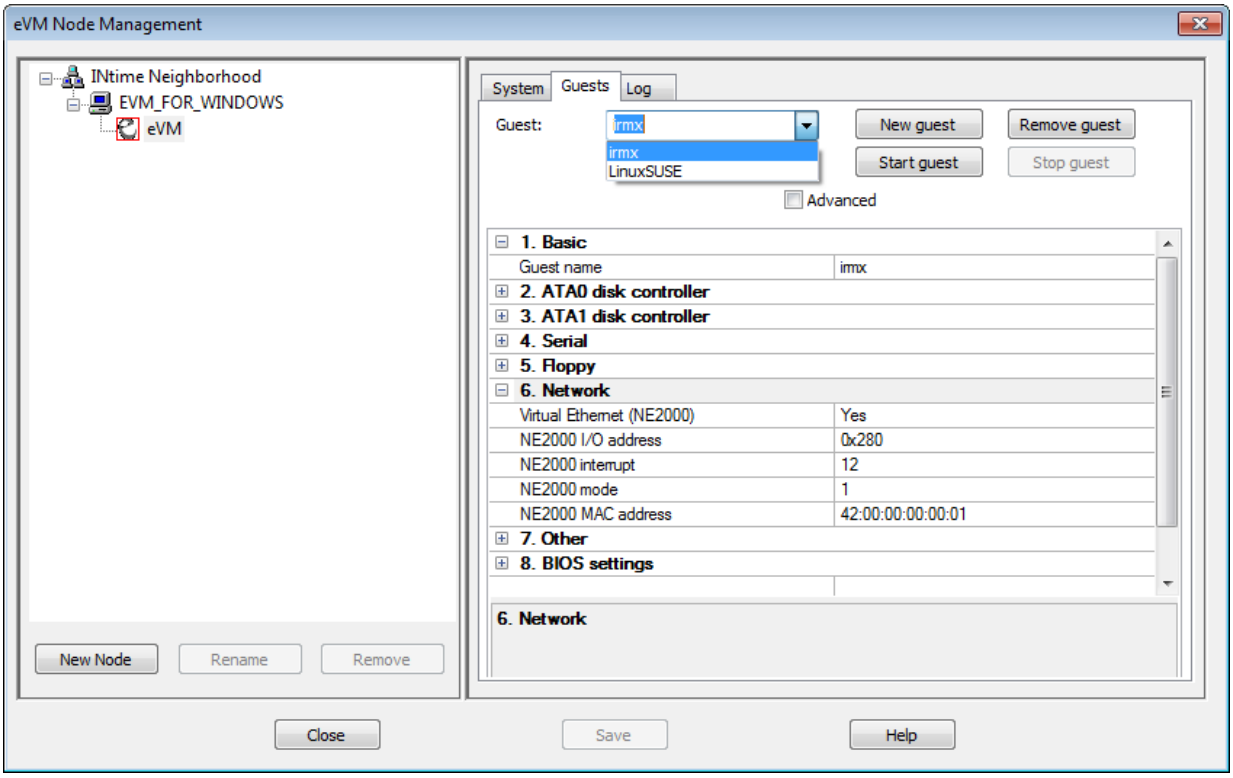


Figure 4.34 - eVM Control Panel, Selecting the Guest OS

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

4. The guest OS, iRMX, will boot in the virtual machine, an eVM Console window will open, and the default debug output from iRMX will be output to the virtual COM device and show up in the terminal emulator window, Figure 4.35.

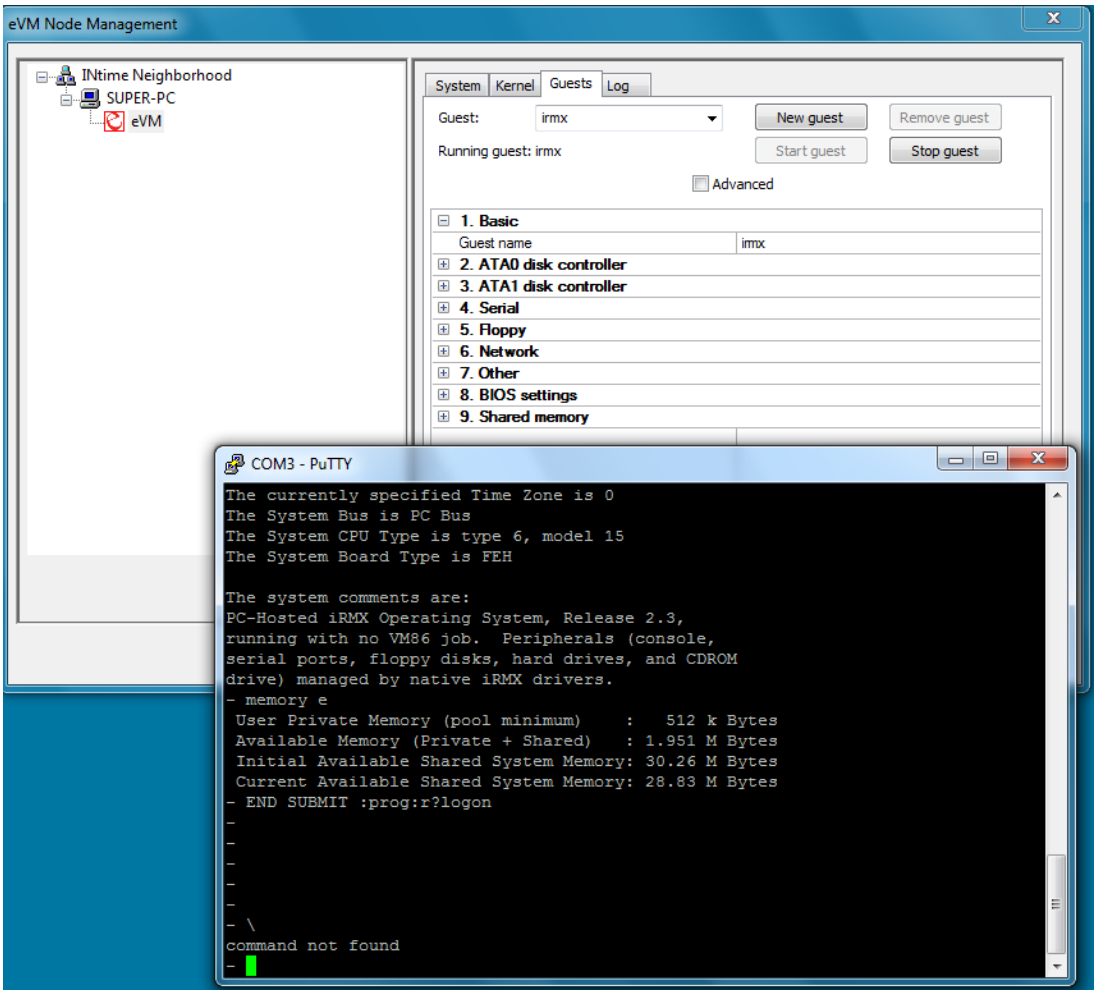


Figure 4.35 - Guest RTOS, iRMX, Booting in the Virtual Machine

Note: for this example we are using the freeware PuTTY application, but any terminal emulation application can be used.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

5. Once iRMX boots, you can now type iRMX system commands in the terminal emulator and iRMX will respond exactly as if it were running on its own dedicated hardware platform.

6. To stop iRMX, in the eVM Control Panel from the toolbar, select the **stop** button (blue square); and the eVM Control Panel will prompt you if you want to stop the guest OS.

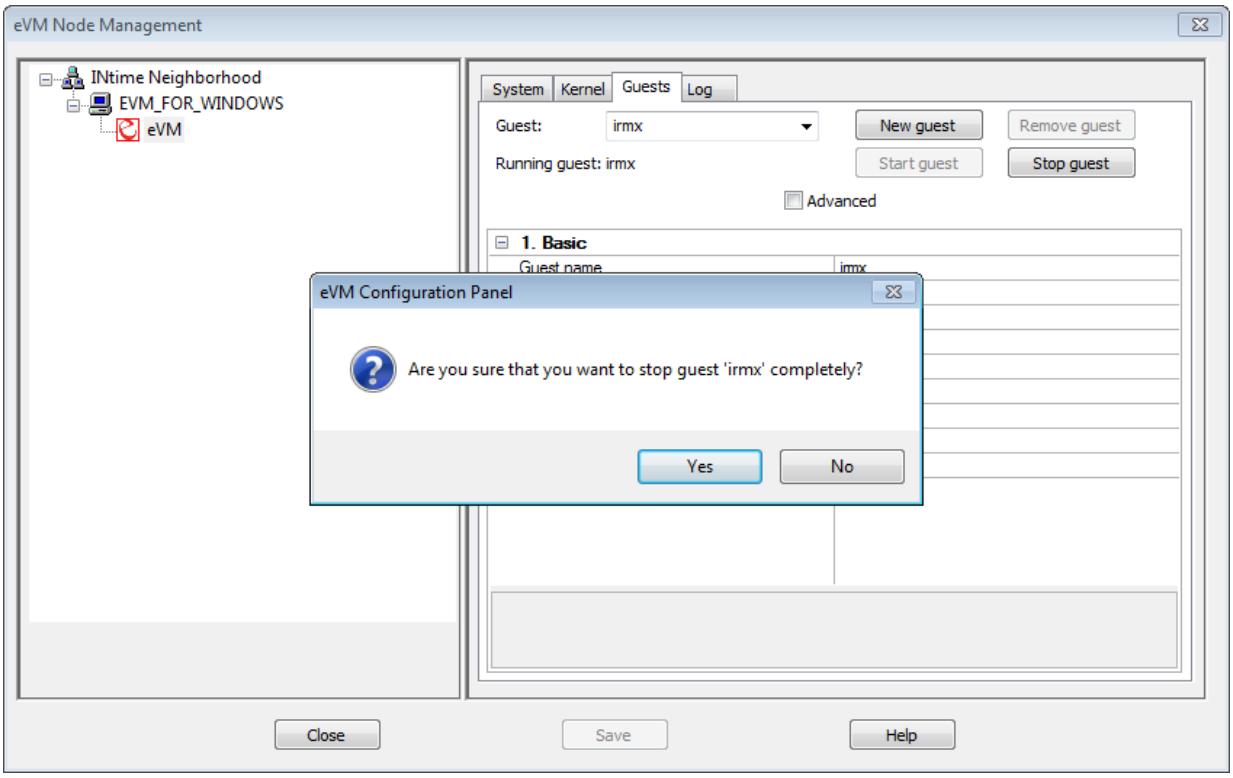


Figure 4.36 - eVM Control Panel, Stopping the Guest OS

7. Click **Yes**, and iRMX will be stopped.

4.4 Installation of a Guest OS

Installing a guest OS on eVM consists of obtaining a disk or image of the target system, and creating a configuration for the guest OS.

The disk image can either be copied from an existing system; or in some cases, it can be installed directly from the installation materials. The configuration is created through the eVM Control Panel utility.

For a list of enabled guest OSes, see Appendix C.

4.4.1 Configure your existing system

By reference to the Virtual Machine specification, Appendix B.2, configure your system in order to match the Virtual Machine hardware interface. The primary point to notice is the absence of

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

a video device and keyboard, so any console usually has to be directed to a serial port by configuration.

4.4.2 Make a disk image of your system

When the system is configured, make a disk image by using a suitable tool. In some cases the disk could be attached to your Windows system and an imaging tool used to make a binary image of the disk. If you are not going to copy the entire drive, make sure you have included the Master Boot Record (MBR) of the disk, as well as, the partition containing your software. There are many Windows tools available to do this, such as the shareware utility, SelfImage.

If you are making an image from a Compact Flash (CF) drive, you may need to adjust the CHS values in the configuration file so that it will boot correctly. Some CF parts have unusual configurations, and these are typically not copied with the data from the drive.

When you have a disk image, copy it to the directory where eVM stores its images. This varies depending on which version of Windows you are using, but the environment variable %EVMCFG% always contains the name of this directory. There is also a link to it in the Start menu. Go to Start->eVM for Windows->eVM Data Folder to open an Explorer Window in this location. Copy your disk image file here.

4.4.3 Make a guest configuration

Now you have a disk image, you need to create a configuration for your system. We will assume that your system has one disk image, and its console is the first serial port (COM1).

1. Open the eVM Control Panel (use the eVM system tray icon or navigate to **Control Panel->eVM Control Panel**).
2. Start a new configuration by clicking on **Guest->New.**
3. In section 1 of the right-hand panel, “Basic”, name your configuration. Configuration names may have up to 12 characters.
4. Assign your disk image. In section 2, change “ATA0 enabled” to **Yes**.
5. In ATA0 master path, click the **…** button at the right and navigate to your disk image and open it.
6. In section 4, “Serial”, change “COM1 access” to “**Windows**”. Note the Windows COM driver name. This is the other end of the virtual serial link from the guest to Windows, and is the device which your terminal emulator will open.
7. In section 8, “BIOS settings”, change “Boot device 1” to **Disk** so the BIOS knows to look for a disk device to boot from.
8. Save your settings (**File->Save**).

You are now ready to test your configuration. Open a terminal emulator and use it to open the serial port you noted in step 6 above.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

Start your guest by clicking on the **Start** button on the toolbar, or by selecting **Guest->Start** from the dropdown.

4.4.4 Installing your operating system from installation materials

In some cases you can install your target operating system using standard installation materials, so long as these do not depend on a video/keyboard console. If you have a floppy boot disk image you should copy this to the eVM Data Folder (%EVMCFG%) and assign the image to the floppy drive of your guest configuration. To do this:

1. Open eVM Control Panel and in the right panel, expand the Floppy device.
2. For *Floppy A enabled* select **Enabled**.
3. In **Floppy A path**, click on the **…** button to the right, navigate to the floppy boot disk image file, and select it.
4. From the dropdown, select **File->Save all** to apply the change.

4.5 Specific steps for certain Guest OSes.

Two example configurations are illustrated here.

4.5.1 Installation and configuration of QNX 6.4

This is an example of customizing a target operating system and creating a disk image.

QNX 6.4 may be installed onto eVM for Windows in the following way: You will need to create a configuration which will boot without the use of the video card. A sample build file is included in this example:

1. Use mkifs to create a boot image from this build file, and then copy the boot image to /.boot.

```
#
# The build file for QNX Neutrino booting on a PC
#
[linker="ntox86-ld -T$QNX_TARGET/x86/lib/nto.link %(h!=0, -Ttext
0x%t%)%(d!=0, -Tdata 0x%d%) -o%o %i %[M -L%^i -uinit_%n -lmod_%n%]"]
[virtual=x86,bios +compress] boot = {
      # Reserve 64k of video memory to handle multiple video cards
      startup-bios -s64k

      # PATH is the *safe* path for executables (confstr(_CS_PATH...))
      # LD_LIBRARY_PATH is the *safe* path for libraries
(confstr(_CS_LIBPATH))
      #    i.e. This is the path searched for libs in setuid/setgid
executables.
      PATH=/proc/boot:/bin:/usr/bin:/opt/bin
LD_LIBRARY_PATH=/proc/boot:/lib:/usr/lib:/lib/dll:/opt/lib procnto-instr
}
```

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

```
[+script] startup-script = {
      display_msg " "
      display_msg "QNX Neutrino 6.4 for eVM"
      display_msg " "

      # To save memory make everyone use the libc in the boot image!
      # For speed (less symbolic lookups) we point to libc.so.3 instead of
libc.so
      procmgr_symlink ../../proc/boot/libc.so.3 /usr/lib/ldqnx.so.2

      display_msg "----> Starting PCI services"
      seedres
      pci-bios
      waitfor /dev/pci

      display_msg "----> Starting console service"
      devc-ser8250 -e -b57600 &
      waitfor /dev/ser1
      reopen /dev/ser1

      # Default user programs to priorty 10, other scheduler (pri=10o)
      # Tell "diskboot" this is a hard disk boot (-b1)
      # Tell "diskboot" to use DMA on IDE drives (-D1)
      # Start 4 text consoles buy passing "-n4" to "devc-con" (-o)
      # By adding "-e" linux ext2 filesystem will be mounted as well.
      #[pri=10o] PATH=/proc/boot diskboot -b1 -D0 -odevc-con,-n4 -odevc-con-
hid,-n4
      display_msg "----> Starting EIDE driver"
      devb-eide blk auto=partition dos exe=all cam quiet eide nobmstr
      waitfor /dev/hd0

      mount /dev/hd0t79 /
      mount -tcd /dev/cd0 /fs/cd0

      display_msg "----> Starting /dev/system/sysinit"
      ksh -c /etc/system/sysinit
}

[type=link] /dev/console=/dev/ser1

# Include the current "libc.so". It will be created as a real file using
# its internal "SONAME", with "libc.so" being a symlink to it. The symlink
# will point to the last "libc.so.*" so if an earlier libc is needed
# (e.g. libc.so.1) add it before the this line.
libc.so
libhiddi.so
libusbdi.so

# Include all the files for the default filesystems
```

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

```
libcam.so
io-blk.so
cam-disk.so
fs-qnx4.so
fs-qnx6.so
fs-dos.so
fs-ext2.so
cam-cdrom.so
fs-cd.so
fs-udf.so

# USB for console driver
devu-ehci.so
devu-ohci.so
devu-uhci.so
devh-usb.so
devh-ps2ser.so

[data=copy]
seedres
pci-bios
devb-eide
devb-umass
devb-ahci
devb-mvSata
devb-adpu320
devb-aha8
umass-enum
[search=${MKIFS_PATH}:${QNX_TARGET}/etc] umass-enum.cfg
io-usb
io-hid
diskboot
slogger
fesh
mount
devc-ser8250
#devc-con
#devc-con-hid

# These files will be unlinked after the system has started
# to release the memory back to the system. They are only
# needed during boot. If needed later on, they will be loaded
# from the boot device.
unlink_list={
/proc/boot/devb-*
}
```

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

2. Now, configure your console device by editing **/etc/config/ttys** and add a line for /dev/ser1:

```
ser1 "/bin/login" qansi-m on
```

3. Configure the network parameters for the virtual Ethernet subnet between Windows and QNX. We are going to define a subnet of 10.100.200.0, and give it two nodes, one for Windows and one for QNX. Edit **rc.local** to load the **ne2000 device** and configure it by adding the following lines:

```
io-pkt-v4 -dne2000 irq=12,ioport=0x280 -ptcpip
ifconfig en0 10.100.200.2 up
inetd &
qconn &
```

4. In Windows, edit the properties of the TenAsys Virtual Ethernet network connection and set the IP address to **10.100.200.1**.

5. Now, make an image of your disk. You can use dd and write the image to another partition, or device (for example):

```
dd if=/dev/hd0 of=/mnt/tmp/qnximg.bin count=9900
```

Note: the size (ifs) parameter must include both the boot sector and the partition that you want to include in the image.

6. Copy the output file to your Windows %EVMCFG% directory and assign it as the first hard drive for your QNX configuration.

7. Make sure that you have at least one virtual serial terminal defined and add it to the configuration.

4.5.2 Installation and configuration of Windows CE

This is an example of installing a guest operating system using the materials provided by the vendor.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

1. Create a DOS-bootable floppy diskette.
2. Copy the following files to it from your Windows CE 6.0 installation:
   loadcepc.exe
   eboot.bin
3. Create an autoexec.bat file with these commands in it:

```
@echo off
verify off
loadcepc /v /e:280:12:192.168.99.2 EBOOT.BIN
```

Note: this will load KITL using the NE2000 virtual Ethernet driver.

4. Configure your Windows Virtual Ethernet driver to use IP address **192.168.99.1**.
5. Now, make a disk image of your floppy diskette, and copy it to the %EVMCFG% directory.
6. Add a floppy device to your configuration and assign this file to it.

Note: this will load the KITL components, so you can now connect to the target using your Windows CE 6.0 installation.

4.6 Driver configuration and setup

Now that we have eVM installed under Windows, the host OS, and have installed a guest OS, we need to dedicate devices to the host OS, to the guest OS, or share them between both. This section will describe how to use eVM Device Manager to accomplish this.

4.6.1 eVM Device Manager

The eVM Device Manager can be launched from the eVM Control Panel by selecting **eVM Device Manager** from the eVM Configuration Panel, Figure 4.37, in Windows Vista or Windows 7 by clicking the **Windows Start button** and entering "***devconfig.exe***" into *Search programs and files* and click on **devconfig.exe**, Figure 4.38, or in Windows XP by selecting **Start->Run**, enter "***devconfig.exe***", and click **OK**, Figure 4.39.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

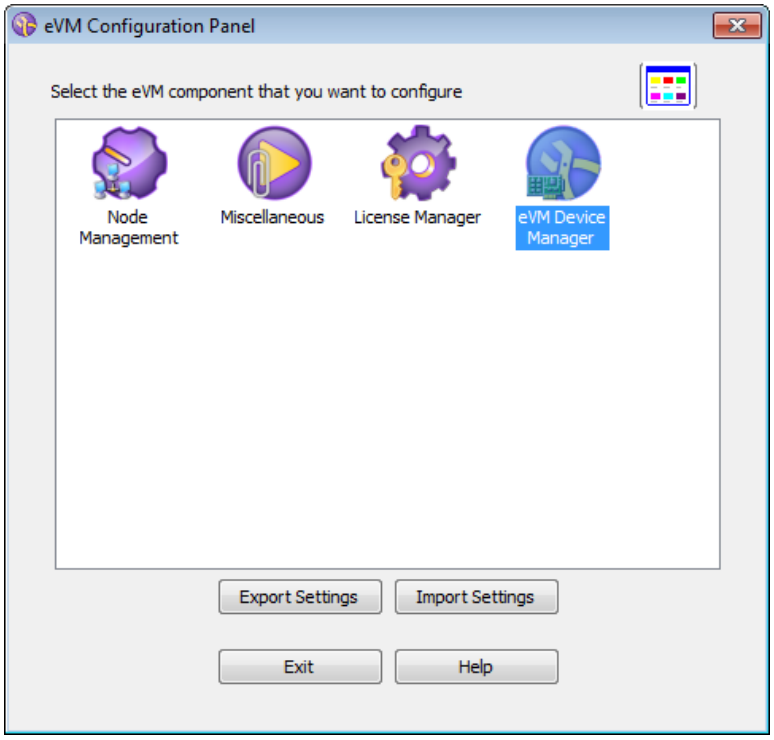


Figure 4.37 – Launching the *eVM Device Manager* from the *eVM Configuration Panel*

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

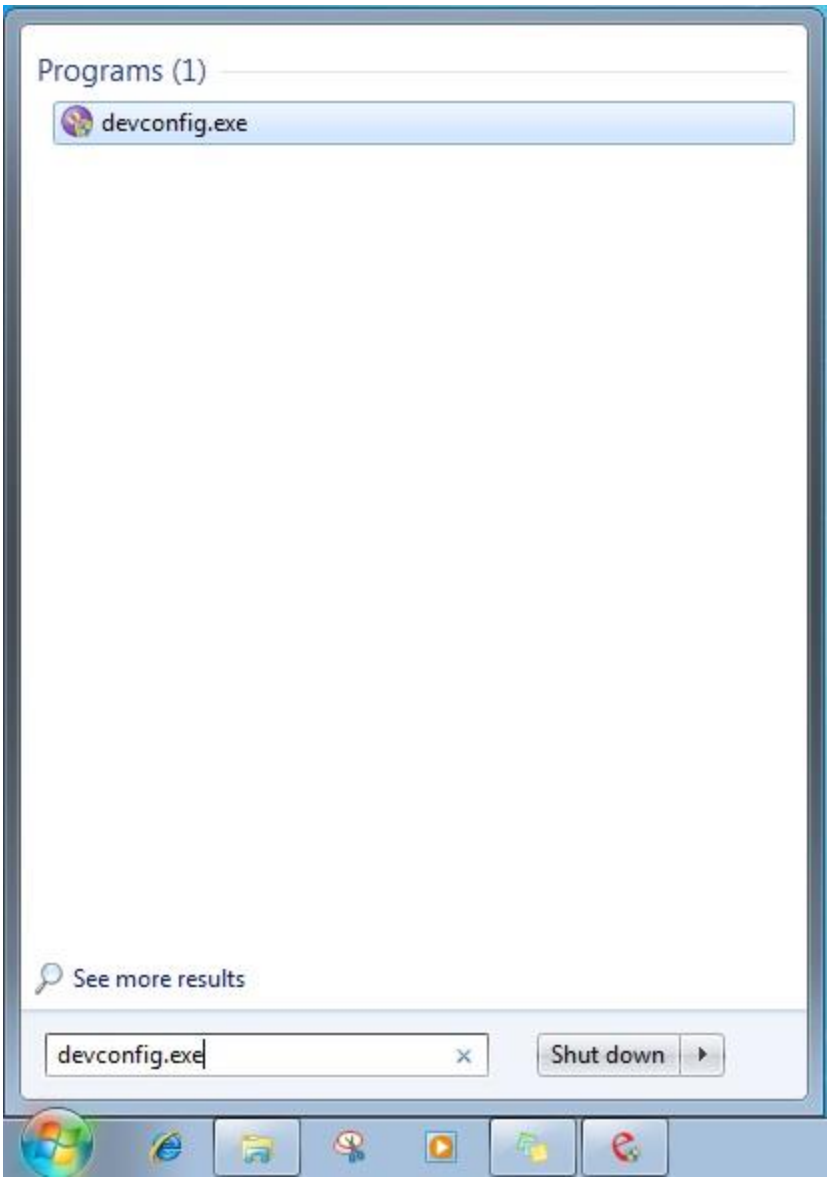


Figure 4.38 - eVM Device Manager launch from Widows Vista or Window 7

Figure 4.39 - eVM Device Manager launch from Windows XP

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

When the eVM Device Manager launches, you will see two active panes, Figure 4.40. The left pane shows two trees of device classes and devices: one controlled by Windows and the other managed by eVM. In either tree you can expand and collapse classes.

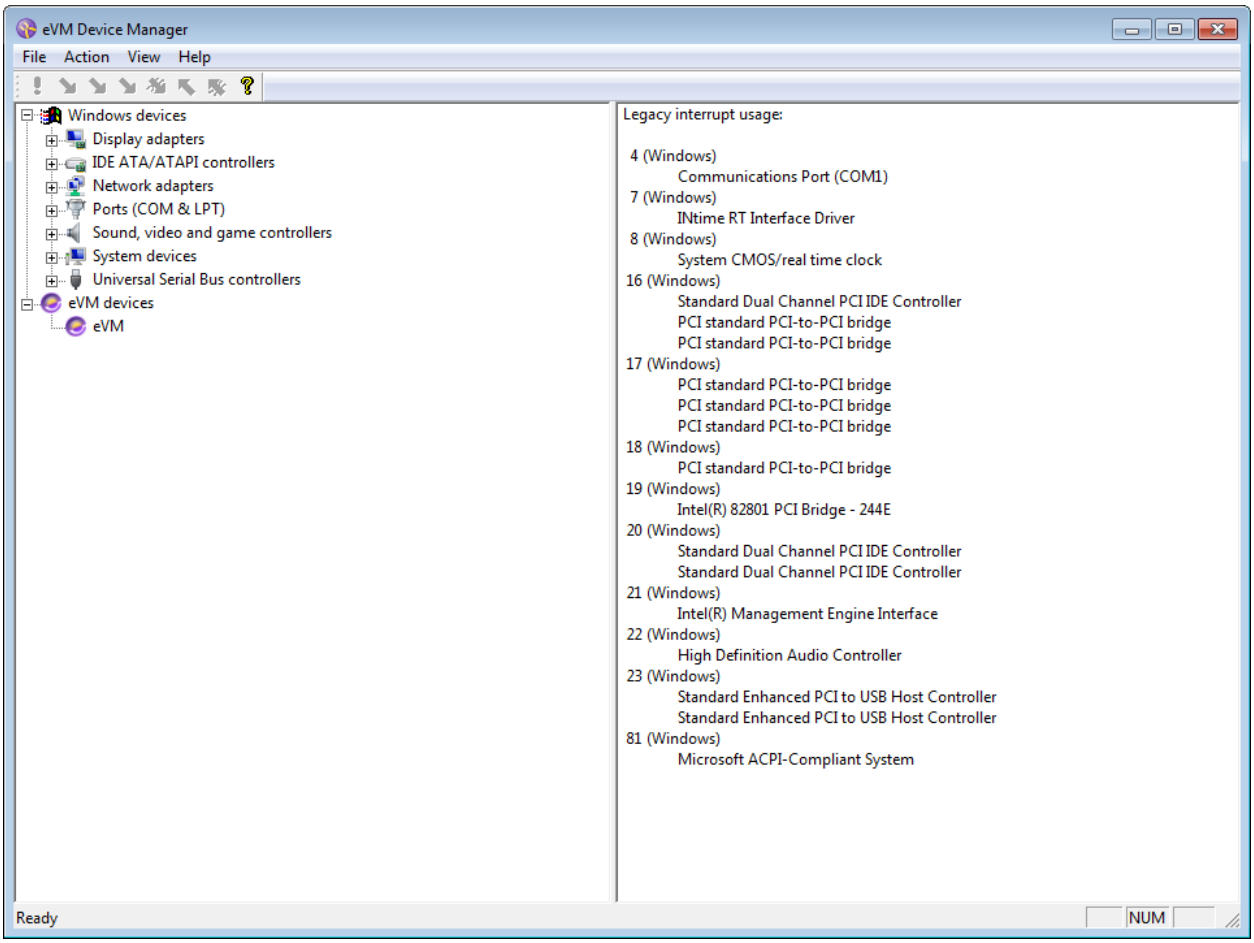


Figure 4.40 - eVM Device Manager showing all the devices that can be passed.

You can use the View menu to select between showing only devices that can be passed or all devices. Figure 4.40 shows eVM Device Manager with only the devices that can be passed being displayed (Default); and when you select the **View** dropdown, you have *All devices* as one of the choices,

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

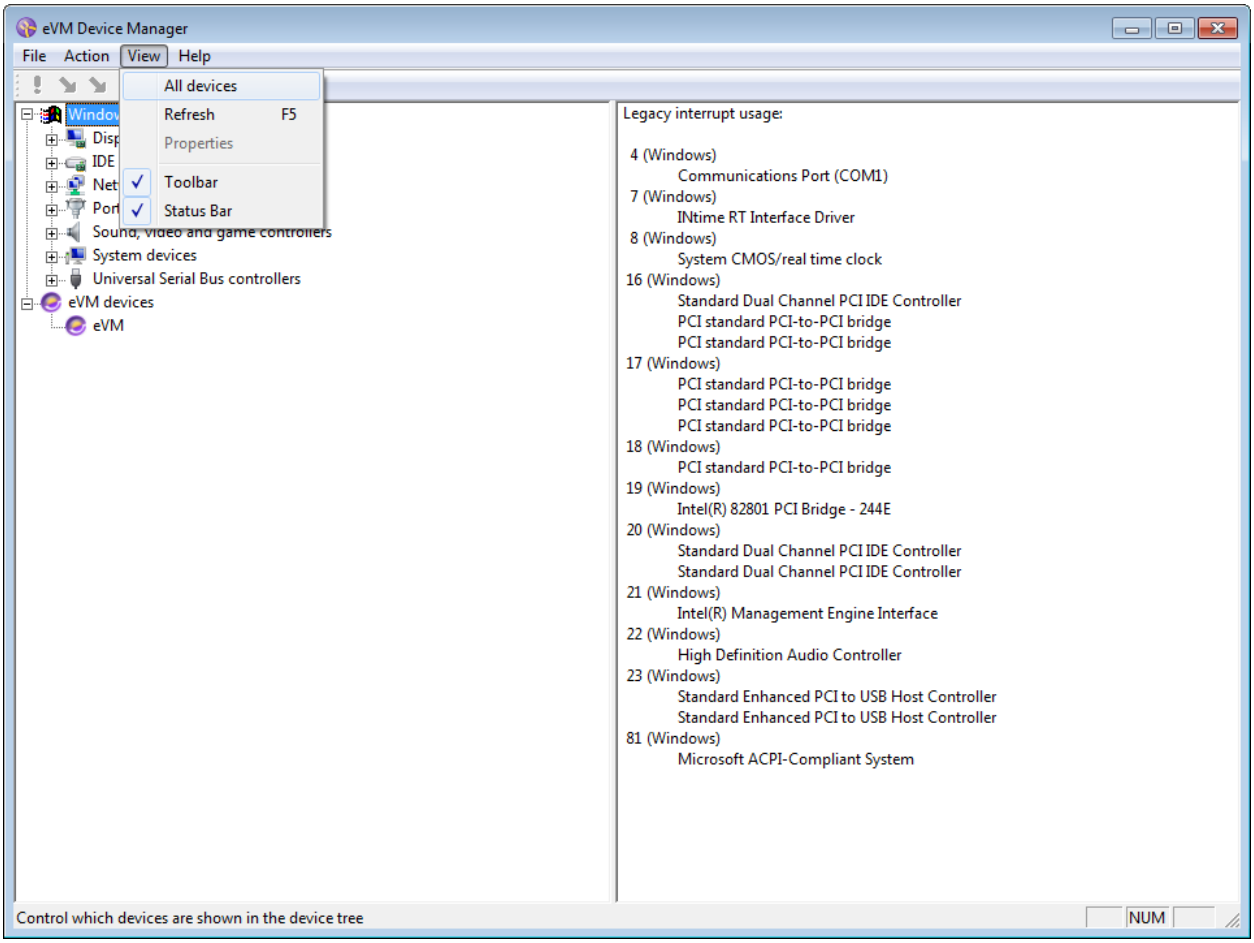


Figure 4.41 - eVM Device Manager – how to change to show all the devices.

Figure 4.42 shows eVM Device Manager with all devices being displayed, and when you select the **View** dropdown, you have *Devices that can be passed* as one of the choices.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

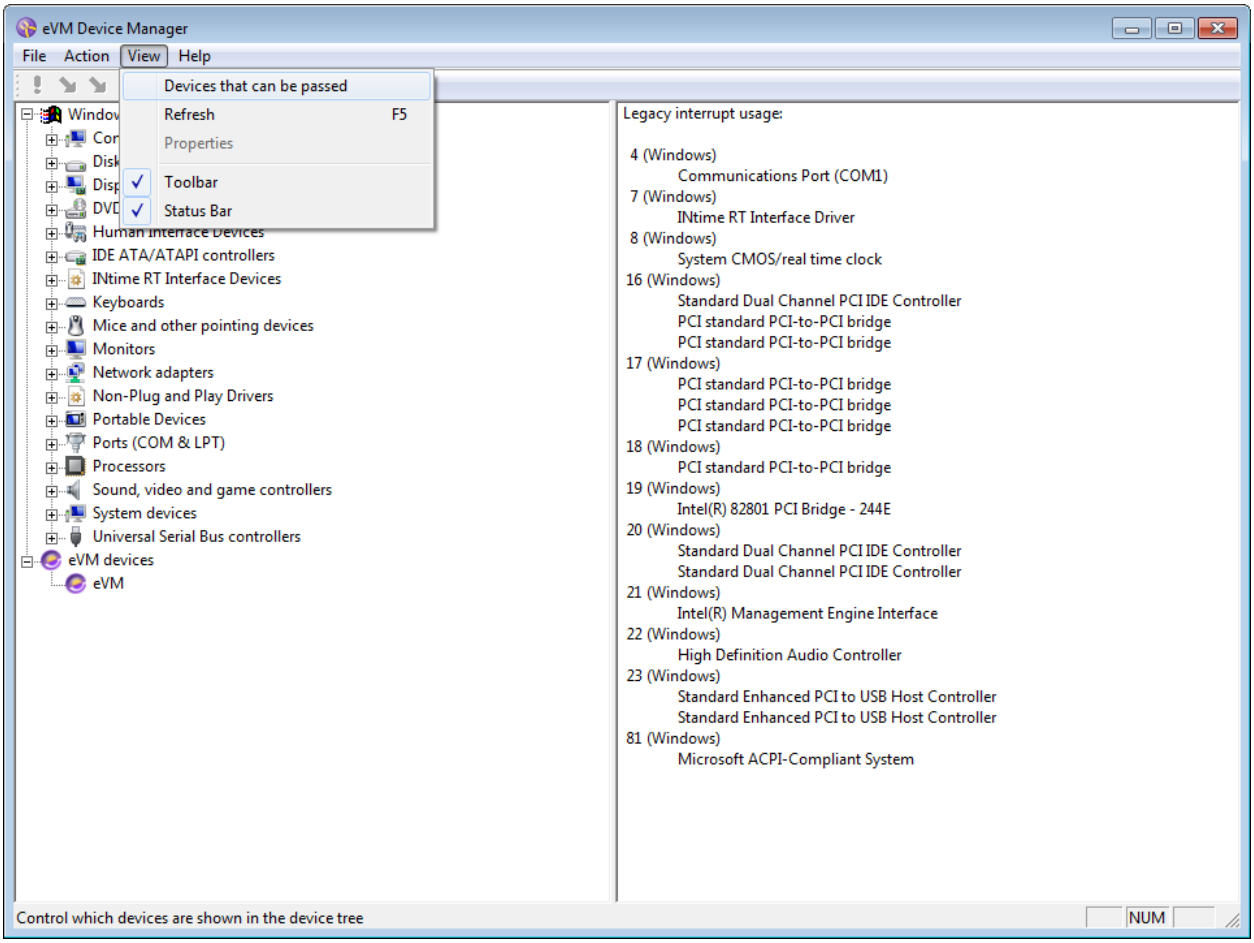


Figure 4.42 - eVM Device Manager Showing All Devices

4.6.2 Passing a Legacy IRQ Device to eVM

For this example, let's choose COM1 as the legacy IRQ device. To confirm that this device fits the criteria, from the dropdown select **View->Devices that can be passed** and look at the right pane and see if COM1 is listed as a device with *Legacy interrupt usage*. Then look at the left pane and expand the Windows devices **Ports (COM & LPT)** device node to see that COM1 is a Windows device that is a device that can be passed to eVM.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

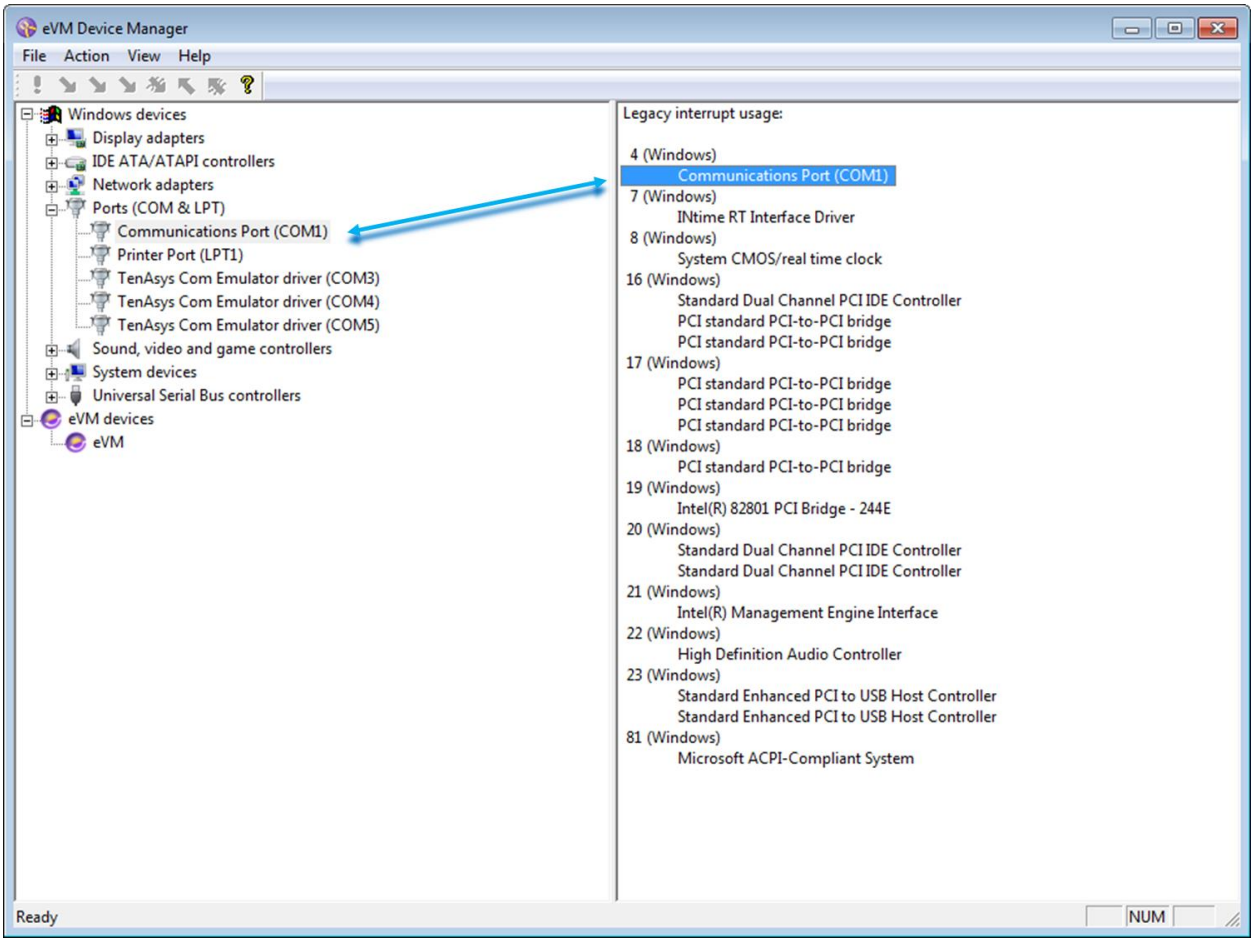


Figure 4.43 - COM1 properties in eVM Device Manager

Figure 4.43 shows that COM1 is a Windows device that can be passed and is a legacy interrupt device.

In the left pane, click on the **Communications Port (COM1)**, and then either click on the **Pass to eVM** button, Figure 4.44.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

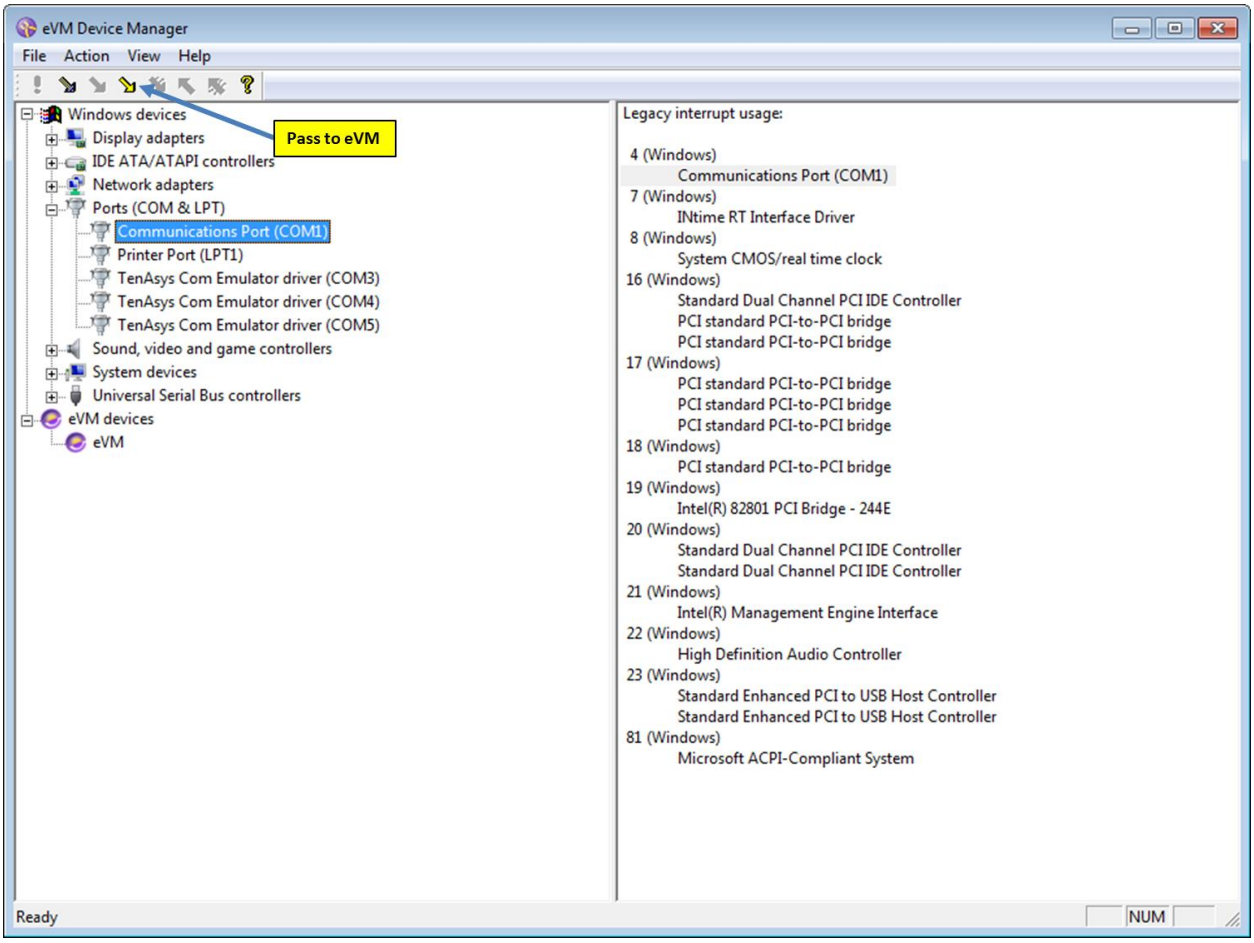


Figure 4.44 - Pass Device Using Pass to eVM Button

Or, from the dropdown, select **Action->Pass to eVM (with legacy IRQ)**, Figure 4.45.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

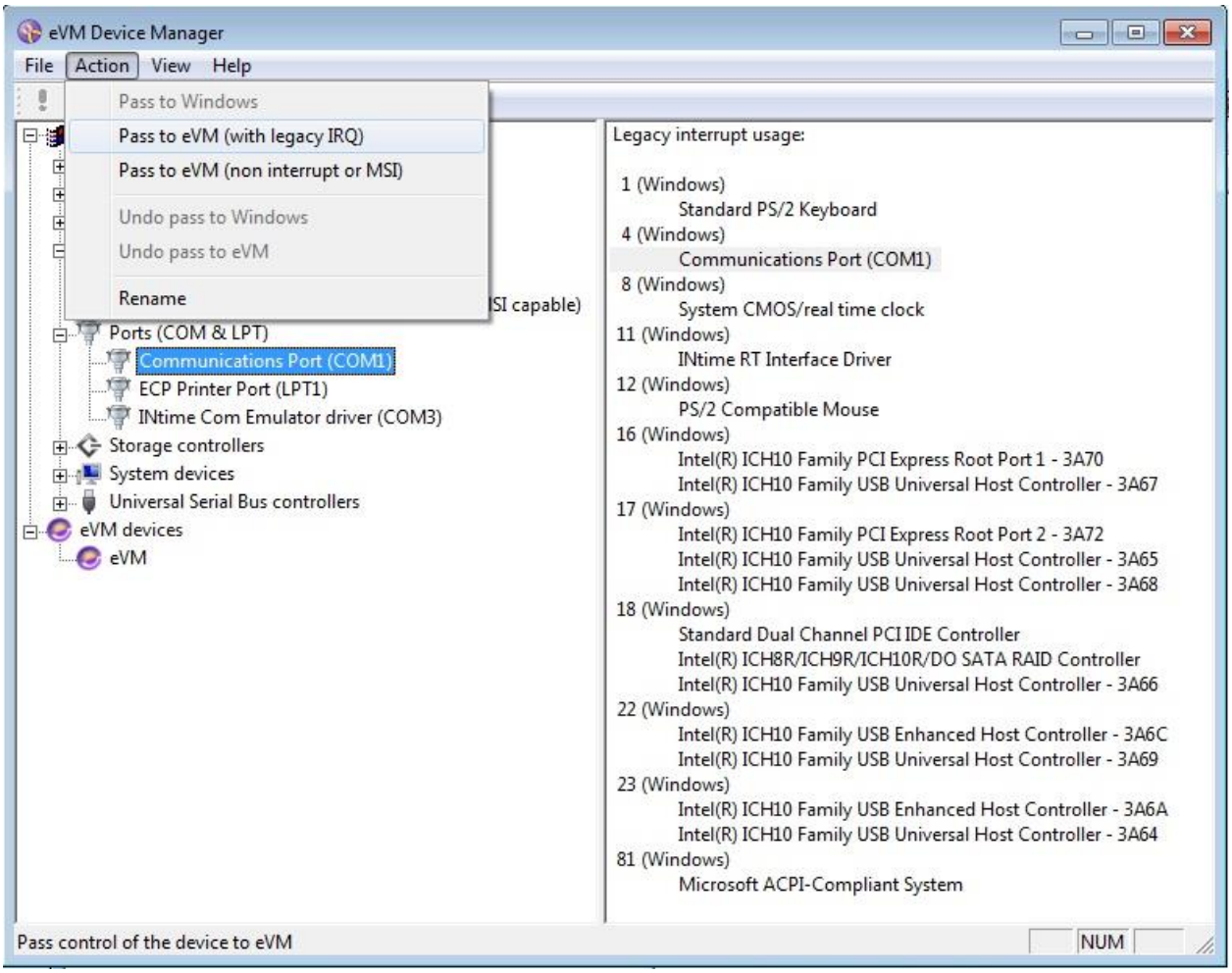

eVM Device Manager
File
Action
View
Help
Pass to Windows
Pass to eVM (with legacy IRQ)
Pass to eVM (non interrupt or MSI)
Undo pass to Windows
Undo pass to eVM
Rename
SI capable)
Ports (COM & LPT)
Communications Port (COM1)
ECP Printer Port (LPT1)
INtime Com Emulator driver (COM3)
Storage controllers
System devices
Universal Serial Bus controllers
eVM devices
eVM
Legacy interrupt usage:
1 (Windows)
Standard PS/2 Keyboard
4 (Windows)
Communications Port (COM1)
8 (Windows)
System CMOS/real time clock
11 (Windows)
INtime RT Interface Driver
12 (Windows)
PS/2 Compatible Mouse
16 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 1 - 3A70
Intel(R) ICH10 Family USB Universal Host Controller - 3A67
17 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 2 - 3A72
Intel(R) ICH10 Family USB Universal Host Controller - 3A65
Intel(R) ICH10 Family USB Universal Host Controller - 3A68
18 (Windows)
Standard Dual Channel PCI IDE Controller
Intel(R) ICH8R/ICH9R/ICH10R/DO SATA RAID Controller
Intel(R) ICH10 Family USB Universal Host Controller - 3A66
22 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6C
Intel(R) ICH10 Family USB Universal Host Controller - 3A69
23 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6A
Intel(R) ICH10 Family USB Universal Host Controller - 3A64
81 (Windows)
Microsoft ACPI-Compliant System
Pass control of the device to eVM
NUM

Figure 4.45 - Pass Device Using Action Dropdown

When you select the device to pass, the right pane will show that you have selected COM1 to be passed, but at this point, the device configuration has not changed, Figure 4.46.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

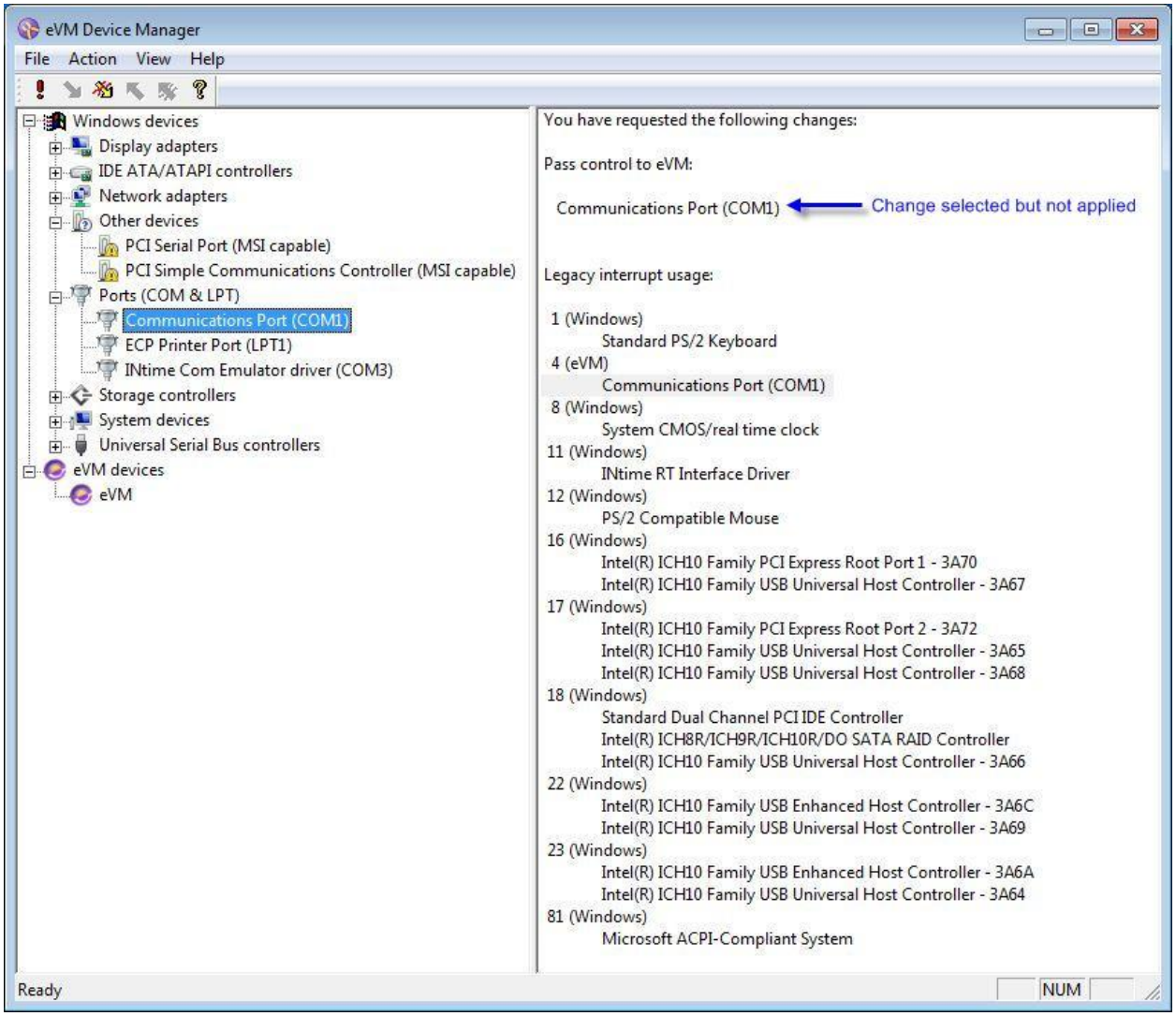

eVM Device Manager
File Action View Help
Windows devices
Display adapters
IDE ATA/ATAPI controllers
Network adapters
Other devices
PCI Serial Port (MSI capable)
PCI Simple Communications Controller (MSI capable)
Ports (COM & LPT)
Communications Port (COM1)
ECP Printer Port (LPT1)
INtime Com Emulator driver (COM3)
Storage controllers
System devices
Universal Serial Bus controllers
eVM devices
eVM
You have requested the following changes:
Pass control to eVM:
Communications Port (COM1)
Change selected but not applied
Legacy interrupt usage:
1 (Windows)
Standard PS/2 Keyboard
4 (eVM)
Communications Port (COM1)
8 (Windows)
System CMOS/real time clock
11 (Windows)
INtime RT Interface Driver
12 (Windows)
PS/2 Compatible Mouse
16 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 1 - 3A70
Intel(R) ICH10 Family USB Universal Host Controller - 3A67
17 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 2 - 3A72
Intel(R) ICH10 Family USB Universal Host Controller - 3A65
Intel(R) ICH10 Family USB Universal Host Controller - 3A68
18 (Windows)
Standard Dual Channel PCI IDE Controller
Intel(R) ICH8R/ICH9R/ICH10R/DO SATA RAID Controller
Intel(R) ICH10 Family USB Universal Host Controller - 3A66
22 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6C
Intel(R) ICH10 Family USB Universal Host Controller - 3A69
23 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6A
Intel(R) ICH10 Family USB Universal Host Controller - 3A64
81 (Windows)
Microsoft ACPI-Compliant System
Ready
NUM

Figure 4.46 - Device Pass Request Selected

To actually apply the requested change in device configuration, you must save the configuration. This is done by either clicking the **Save configuration** button, Figure 4.47.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

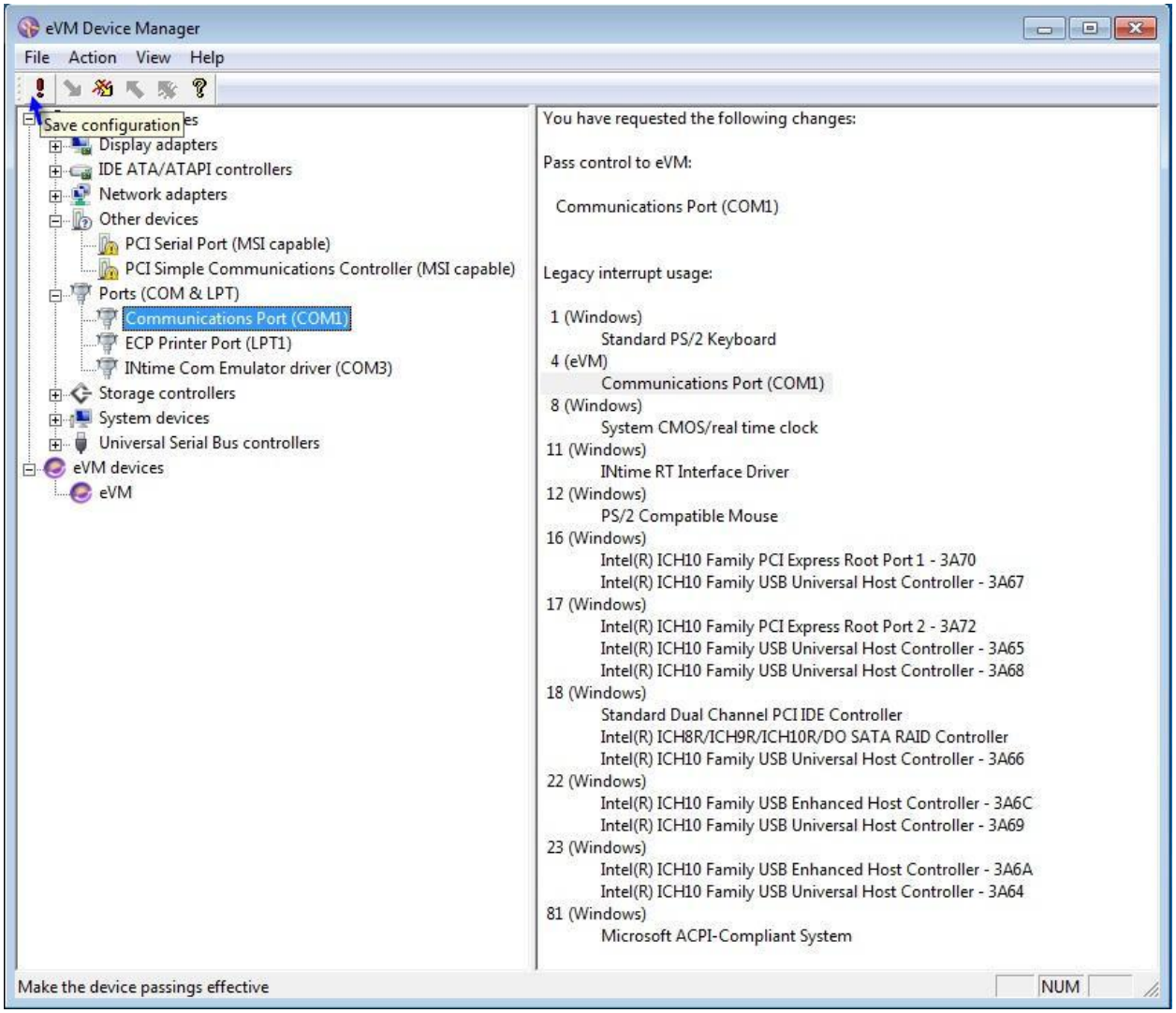

eVM Device Manager
File Action View Help
Save configuration
Display adapters
IDE ATA/ATAPI controllers
Network adapters
Other devices
PCI Serial Port (MSI capable)
PCI Simple Communications Controller (MSI capable)
Ports (COM & LPT)
Communications Port (COM1)
ECP Printer Port (LPT1)
INtime Com Emulator driver (COM3)
Storage controllers
System devices
Universal Serial Bus controllers
eVM devices
eVM
You have requested the following changes:
Pass control to eVM:
Communications Port (COM1)
Legacy interrupt usage:
1 (Windows)
Standard PS/2 Keyboard
4 (eVM)
Communications Port (COM1)
8 (Windows)
System CMOS/real time clock
11 (Windows)
INtime RT Interface Driver
12 (Windows)
PS/2 Compatible Mouse
16 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 1 - 3A70
Intel(R) ICH10 Family USB Universal Host Controller - 3A67
17 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 2 - 3A72
Intel(R) ICH10 Family USB Universal Host Controller - 3A65
Intel(R) ICH10 Family USB Universal Host Controller - 3A68
18 (Windows)
Standard Dual Channel PCI IDE Controller
Intel(R) ICH8R/ICH9R/ICH10R/DO SATA RAID Controller
Intel(R) ICH10 Family USB Universal Host Controller - 3A66
22 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6C
Intel(R) ICH10 Family USB Universal Host Controller - 3A69
23 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6A
Intel(R) ICH10 Family USB Universal Host Controller - 3A64
81 (Windows)
Microsoft ACPI-Compliant System
Make the device passings effective
NUM

Figure 4.47 - Using the Save Configuration Button

Or, from the dropdown select **File->Save the configuration**, Figure 4.48.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

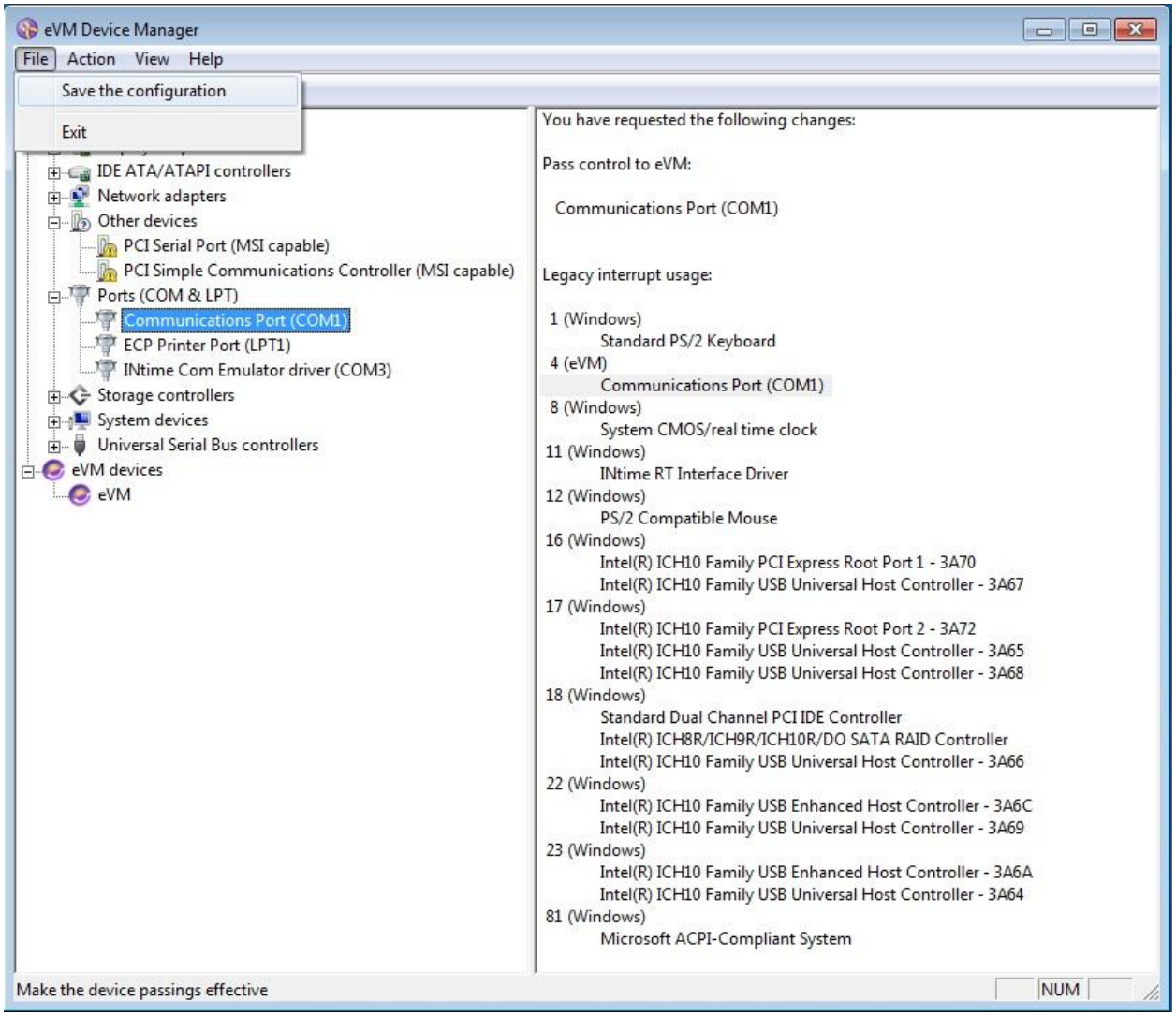

eVM Device Manager
File
Action
View
Help
Save the configuration
Exit
IDE ATA/ATAPI controllers
Network adapters
Other devices
PCI Serial Port (MSI capable)
PCI Simple Communications Controller (MSI capable)
Ports (COM & LPT)
Communications Port (COM1)
ECP Printer Port (LPT1)
INtime Com Emulator driver (COM3)
Storage controllers
System devices
Universal Serial Bus controllers
eVM devices
eVM
You have requested the following changes:
Pass control to eVM:
Communications Port (COM1)
Legacy interrupt usage:
1 (Windows)
Standard PS/2 Keyboard
4 (eVM)
Communications Port (COM1)
8 (Windows)
System CMOS/real time clock
11 (Windows)
INtime RT Interface Driver
12 (Windows)
PS/2 Compatible Mouse
16 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 1 - 3A70
Intel(R) ICH10 Family USB Universal Host Controller - 3A67
17 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 2 - 3A72
Intel(R) ICH10 Family USB Universal Host Controller - 3A65
Intel(R) ICH10 Family USB Universal Host Controller - 3A68
18 (Windows)
Standard Dual Channel PCI IDE Controller
Intel(R) ICH8R/ICH9R/ICH10R/DO SATA RAID Controller
Intel(R) ICH10 Family USB Universal Host Controller - 3A66
22 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6C
Intel(R) ICH10 Family USB Universal Host Controller - 3A69
23 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6A
Intel(R) ICH10 Family USB Universal Host Controller - 3A64
81 (Windows)
Microsoft ACPI-Compliant System
Make the device passings effective
NUM

Figure 4.48 - Using Dropdown to Save the Configuration

Prior to changing the configuration, the *eVM Device Manager* will warn you, if the device that you are passing is a bus master, that your platform needs VT-d support for this device to support DMA transfer. Otherwise you will have to modify the device driver accordingly. See Figure 4.49

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

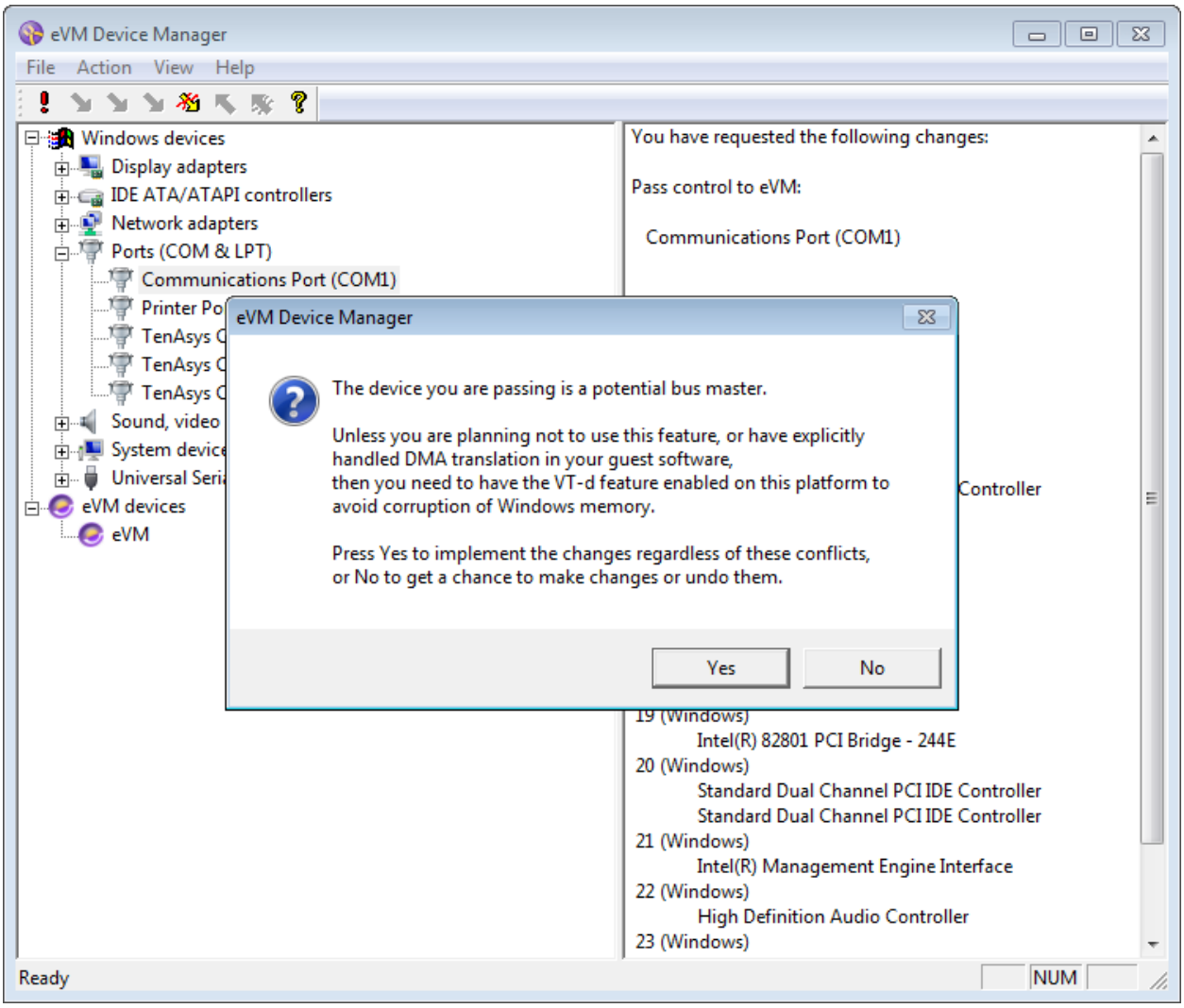

eVM Device Manager
File Action View Help
Windows devices
Display adapters
IDE ATA/ATAPI controllers
Network adapters
Ports (COM & LPT)
Communications Port (COM1)
Printer Po
TenAsys C
TenAsys C
TenAsys C
Sound, video
System device
Universal Seri
eVM devices
eVM
You have requested the following changes:
Pass control to eVM:
Communications Port (COM1)
eVM Device Manager
The device you are passing is a potential bus master.
Unless you are planning not to use this feature, or have explicitly handled DMA translation in your guest software, then you need to have the VT-d feature enabled on this platform to avoid corruption of Windows memory.
Press Yes to implement the changes regardless of these conflicts, or No to get a chance to make changes or undo them.
Yes
No
Controller
19 (Windows)
Intel(R) 82801 PCI Bridge - 244E
20 (Windows)
Standard Dual Channel PCI IDE Controller
Standard Dual Channel PCI IDE Controller
21 (Windows)
Intel(R) Management Engine Interface
22 (Windows)
High Definition Audio Controller
23 (Windows)
Ready
NUM

Figure 4.49 Warning that you need VT-d support if the device is a bus master and you want to DMA to it.

Next, the device configuration will be accepted, but it will not actually be applied until the platform has been rebooted. eVM Device Manager prompts you to restart the platform, Figure 4.50.

Note: This is Windows requirement and not all systems will require a *Restart* for the configuration change to take effect. See *Windows Help* on Device Configuration for more information.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

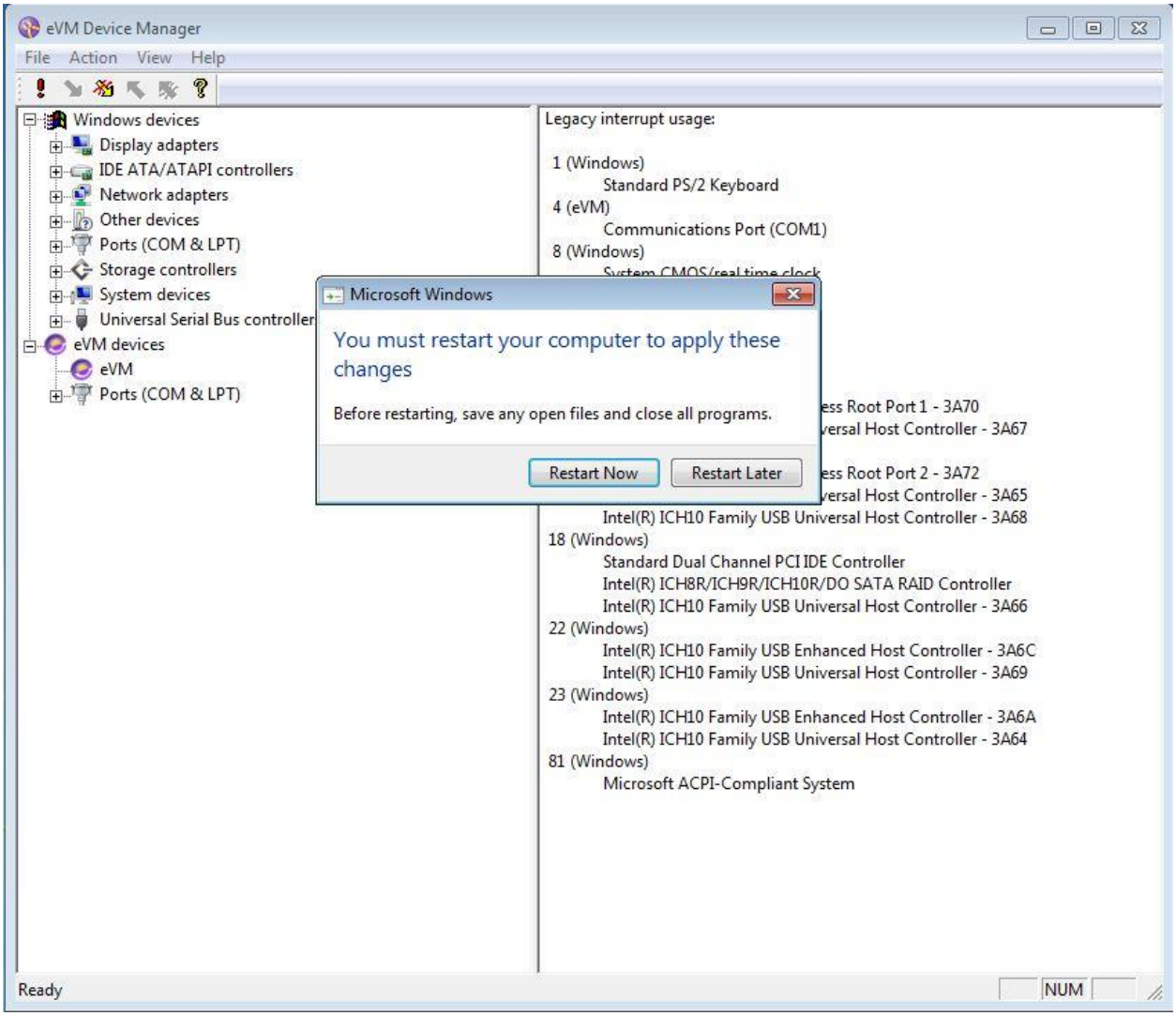


Figure 4.50 - Platform Restart Required to Apply New Device Configuration

When the platform has been restarted, launch eVM Device Manager; and expand the **Ports (COM & LPT)** node under *Windows devices* and under *eVM devices,* Figure 4.51. You will see that the Communications Port (COM1) is no longer listed as a Windows device, but it is now listed as an eVM device.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

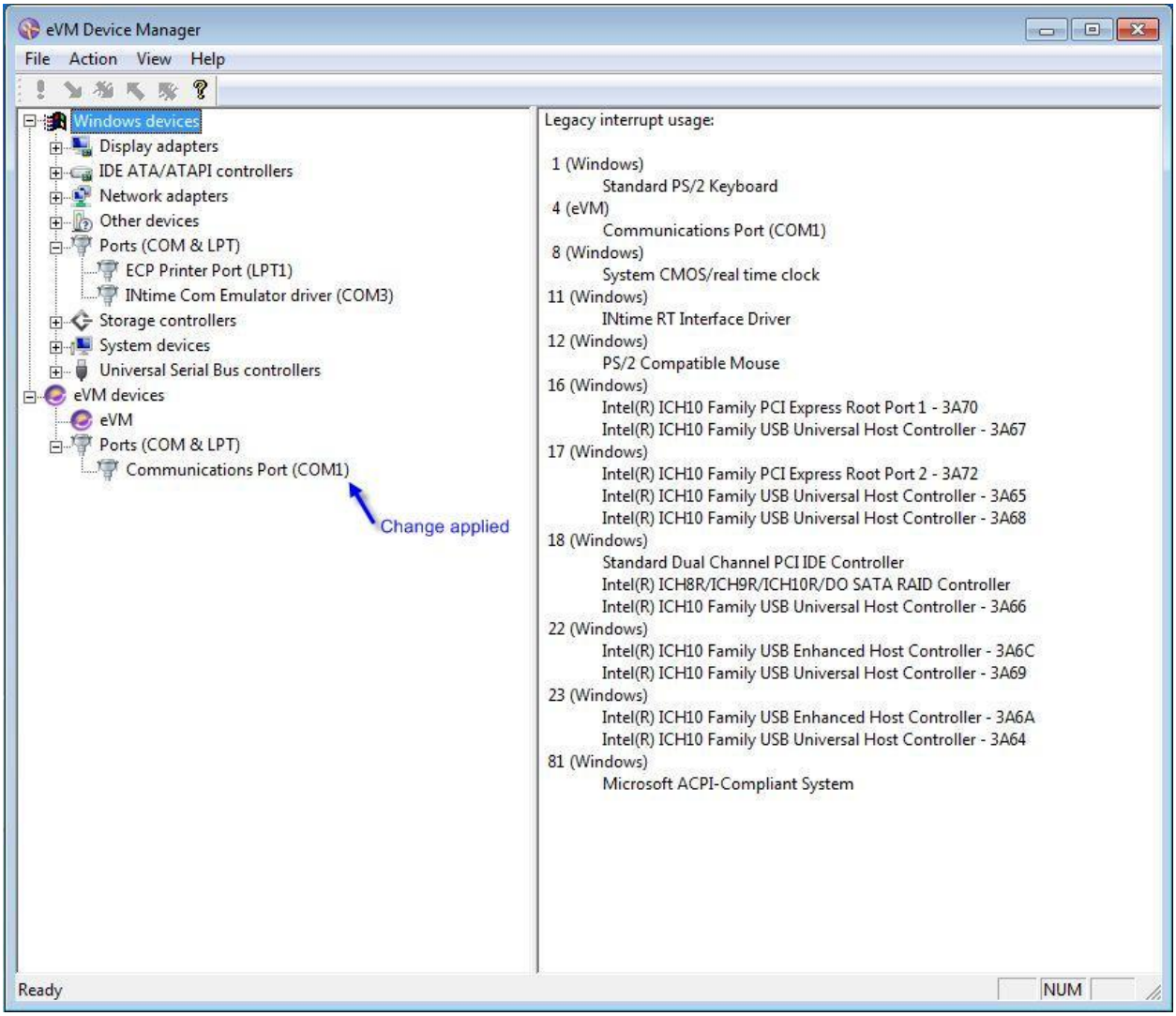

eVM Device Manager
File Action View Help
Windows devices
Display adapters
IDE ATA/ATAPI controllers
Network adapters
Other devices
Ports (COM & LPT)
ECP Printer Port (LPT1)
INtime Com Emulator driver (COM3)
Storage controllers
System devices
Universal Serial Bus controllers
eVM devices
eVM
Ports (COM & LPT)
Communications Port (COM1)
Change applied
Legacy interrupt usage:
1 (Windows)
Standard PS/2 Keyboard
4 (eVM)
Communications Port (COM1)
8 (Windows)
System CMOS/real time clock
11 (Windows)
INtime RT Interface Driver
12 (Windows)
PS/2 Compatible Mouse
16 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 1 - 3A70
Intel(R) ICH10 Family USB Universal Host Controller - 3A67
17 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 2 - 3A72
Intel(R) ICH10 Family USB Universal Host Controller - 3A65
Intel(R) ICH10 Family USB Universal Host Controller - 3A68
18 (Windows)
Standard Dual Channel PCI IDE Controller
Intel(R) ICH8R/ICH9R/ICH10R/DO SATA RAID Controller
Intel(R) ICH10 Family USB Universal Host Controller - 3A66
22 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6C
Intel(R) ICH10 Family USB Universal Host Controller - 3A69
23 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6A
Intel(R) ICH10 Family USB Universal Host Controller - 3A64
81 (Windows)
Microsoft ACPI-Compliant System
Ready
NUM

Figure 4.51 - Device Configuration Change Applied

The legacy IRQ COM1 device has now been passed to eVM for exclusive use by the guest OS.

4.6.3 Conflict Example when Passing Legacy IRQ Device to eVM

Remember the issue with sharing an interrupt between Windows and the guest OS? This would make the guest OS dependent on a Windows interrupt handler running to maintain a deterministic system. It is not allowed to assign a device to the guest and a device to Windows that share a common interrupt. The eVM Device Manager will flag an attempt to do this as a conflict.

To illustrate this, open the eVM Device Manager; and under *Windows devices*, expand **Universal Serial Bus controllers**, Figure 4.52.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

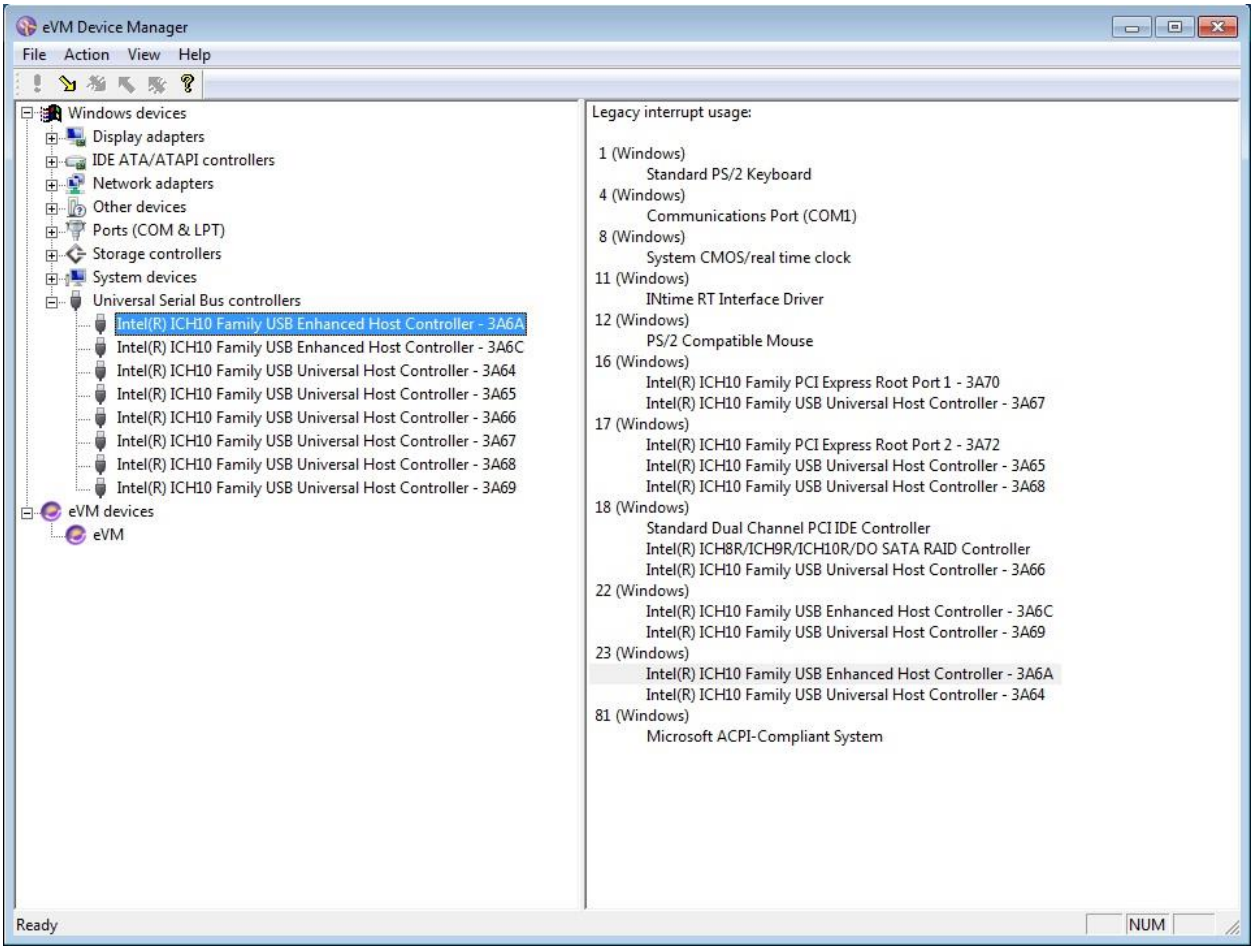

eVM Device Manager
File Action View Help
Windows devices
Display adapters
IDE ATA/ATAPI controllers
Network adapters
Other devices
Ports (COM & LPT)
Storage controllers
System devices
Universal Serial Bus controllers
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6A
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6C
Intel(R) ICH10 Family USB Universal Host Controller - 3A64
Intel(R) ICH10 Family USB Universal Host Controller - 3A65
Intel(R) ICH10 Family USB Universal Host Controller - 3A66
Intel(R) ICH10 Family USB Universal Host Controller - 3A67
Intel(R) ICH10 Family USB Universal Host Controller - 3A68
Intel(R) ICH10 Family USB Universal Host Controller - 3A69
eVM devices
eVM
Legacy interrupt usage:
1 (Windows)
Standard PS/2 Keyboard
4 (Windows)
Communications Port (COM1)
8 (Windows)
System CMOS/real time clock
11 (Windows)
INtime RT Interface Driver
12 (Windows)
PS/2 Compatible Mouse
16 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 1 - 3A70
Intel(R) ICH10 Family USB Universal Host Controller - 3A67
17 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 2 - 3A72
Intel(R) ICH10 Family USB Universal Host Controller - 3A65
Intel(R) ICH10 Family USB Universal Host Controller - 3A68
18 (Windows)
Standard Dual Channel PCI IDE Controller
Intel(R) ICH8R/ICH9R/ICH10R/DO SATA RAID Controller
Intel(R) ICH10 Family USB Universal Host Controller - 3A66
22 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6C
Intel(R) ICH10 Family USB Universal Host Controller - 3A69
23 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6A
Intel(R) ICH10 Family USB Universal Host Controller - 3A64
81 (Windows)
Microsoft ACPI-Compliant System
Ready
NUM

Figure 4.52 - eVM Device Manager Showing Windows USB Controllers

In the left pane, select the first **USB Enhanced Host Controller** and then, from the dropdown, select **Pass to eVM**.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

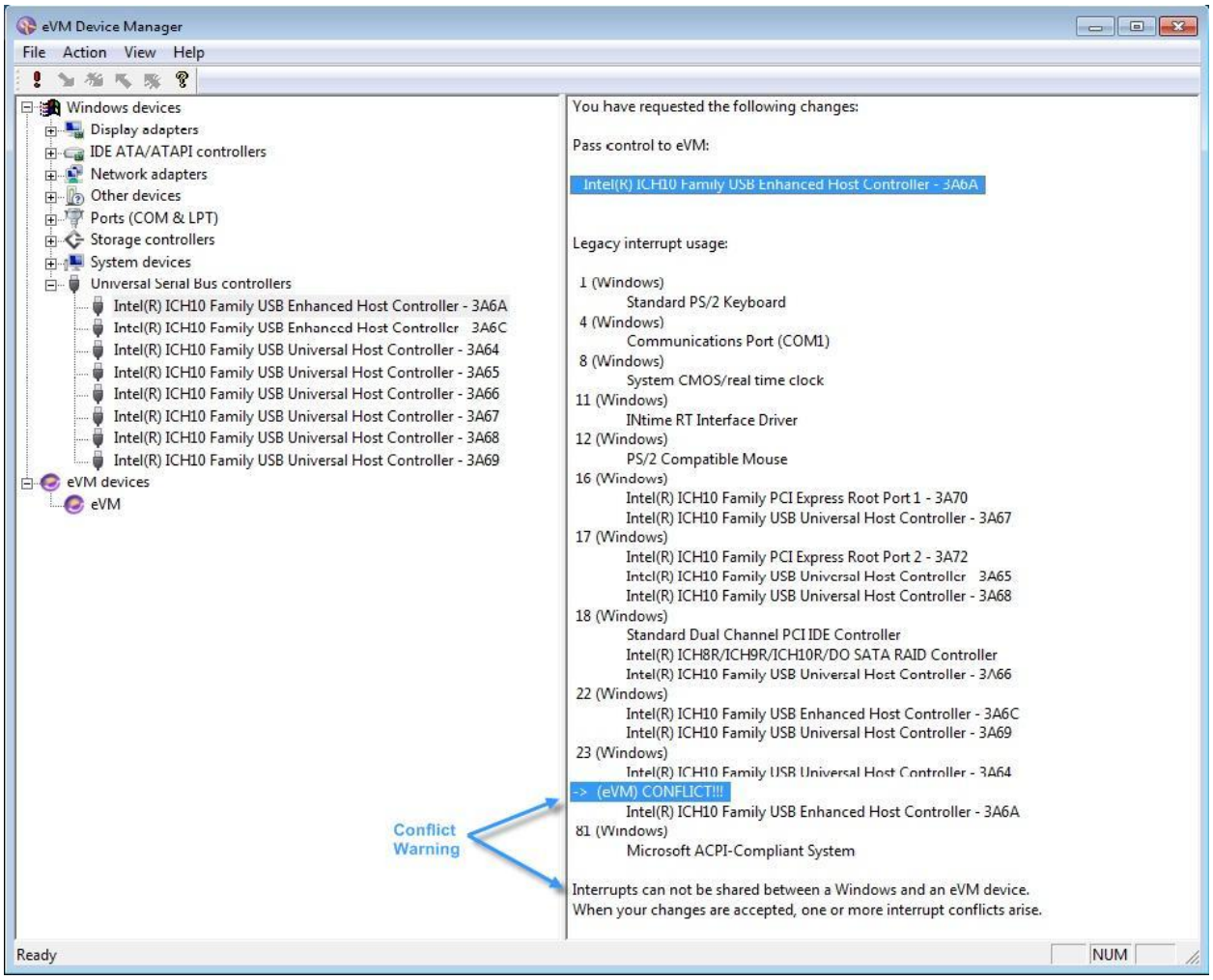

eVM Device Manager
File Action View Help
Windows devices
Display adapters
IDE ATA/ATAPI controllers
Network adapters
Other devices
Ports (COM & LPT)
Storage controllers
System devices
Universal Serial Bus controllers
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6A
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6C
Intel(R) ICH10 Family USB Universal Host Controller - 3A64
Intel(R) ICH10 Family USB Universal Host Controller - 3A65
Intel(R) ICH10 Family USB Universal Host Controller - 3A66
Intel(R) ICH10 Family USB Universal Host Controller - 3A67
Intel(R) ICH10 Family USB Universal Host Controller - 3A68
Intel(R) ICH10 Family USB Universal Host Controller - 3A69
eVM devices
eVM
You have requested the following changes:
Pass control to eVM:
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6A
Legacy interrupt usage:
1 (Windows)
Standard PS/2 Keyboard
4 (Windows)
Communications Port (COM1)
8 (Windows)
System CMOS/real time clock
11 (Windows)
INtime RT Interface Driver
12 (Windows)
PS/2 Compatible Mouse
16 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 1 - 3A70
Intel(R) ICH10 Family USB Universal Host Controller - 3A67
17 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 2 - 3A72
Intel(R) ICH10 Family USB Universal Host Controller - 3A65
Intel(R) ICH10 Family USB Universal Host Controller - 3A68
18 (Windows)
Standard Dual Channel PCI IDE Controller
Intel(R) ICH8R/ICH9R/ICH10R/DO SATA RAID Controller
Intel(R) ICH10 Family USB Universal Host Controller - 3A66
22 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6C
Intel(R) ICH10 Family USB Universal Host Controller - 3A69
23 (Windows)
Intel(R) ICH10 Family USB Universal Host Controller - 3A64
-> (eVM) CONFLICT!!!
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6A
81 (Windows)
Microsoft ACPI-Compliant System
Interrupts can not be shared between a Windows and an eVM device.
When your changes are accepted, one or more interrupt conflicts arise.
Conflict Warning
Ready
NUM

Figure 4.53 - eVM Device Manager Flagging a Conflict

In the right pane, see the conflict warning that is posted by eVM Device Manager, Figure 4.53.

4.6.4 Passing an MSI Device to eVM

In our example system, we have two Ethernet devices. One is an onboard controller that Windows is using, and the other is a PCI Express adapter with MSI capability that can be dedicated to eVM for the guest OS to use. If we expand the Windows devices **Network adapters** node in the left pane of this system, we will see the *Intel® Gigabit CT Desktop Adapter (MSI capable)* device. To illustrate another way to pass a device to eVM, right-click on the **Intel® Gigabit CT Desktop Adapter (MSI capable)**device, and select **Pass to eVM (non interrupt or MSI)**,Figure 4.54.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

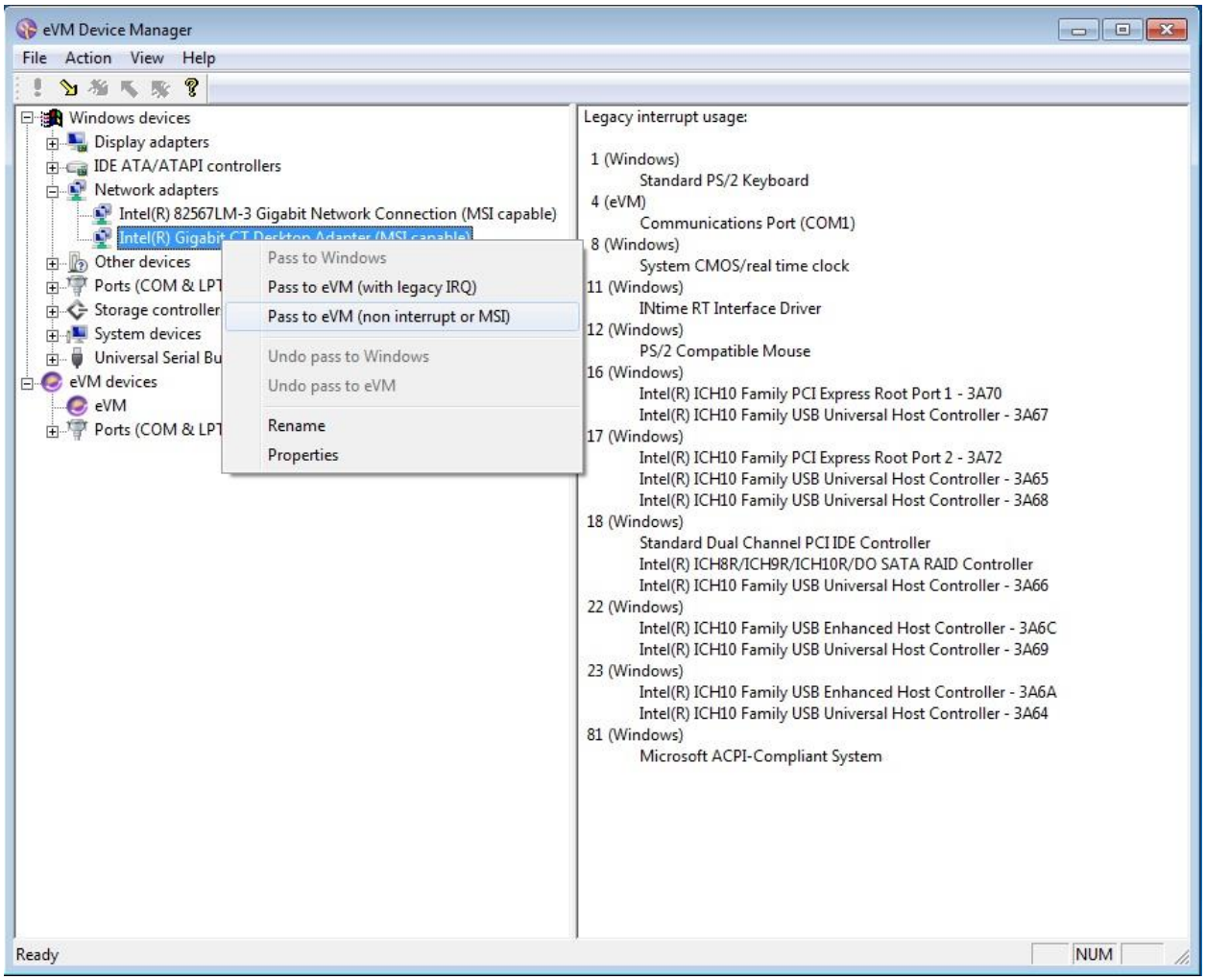

eVM Device Manager
File Action View Help
Windows devices
Display adapters
IDE ATA/ATAPI controllers
Network adapters
Intel(R) 82567LM-3 Gigabit Network Connection (MSI capable)
Other devices
Ports (COM & LPT
Storage controller
System devices
Universal Serial Bu
eVM devices
eVM
Ports (COM & LPT
Pass to Windows
Pass to eVM (with legacy IRQ)
Pass to eVM (non interrupt or MSI)
Undo pass to Windows
Undo pass to eVM
Rename
Properties
Legacy interrupt usage:
1 (Windows)
Standard PS/2 Keyboard
4 (eVM)
Communications Port (COM1)
8 (Windows)
System CMOS/real time clock
11 (Windows)
INtime RT Interface Driver
12 (Windows)
PS/2 Compatible Mouse
16 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 1 - 3A70
Intel(R) ICH10 Family USB Universal Host Controller - 3A67
17 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 2 - 3A72
Intel(R) ICH10 Family USB Universal Host Controller - 3A65
Intel(R) ICH10 Family USB Universal Host Controller - 3A68
18 (Windows)
Standard Dual Channel PCI IDE Controller
Intel(R) ICH8R/ICH9R/ICH10R/DO SATA RAID Controller
Intel(R) ICH10 Family USB Universal Host Controller - 3A66
22 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6C
Intel(R) ICH10 Family USB Universal Host Controller - 3A69
23 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6A
Intel(R) ICH10 Family USB Universal Host Controller - 3A64
81 (Windows)
Microsoft ACPI-Compliant System
Ready
NUM

Figure 4.54 - Right-Click a Device to Pass it to eVM

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

Again, we will see the request to pass the device now listed in the right pane, Figure 4.55.

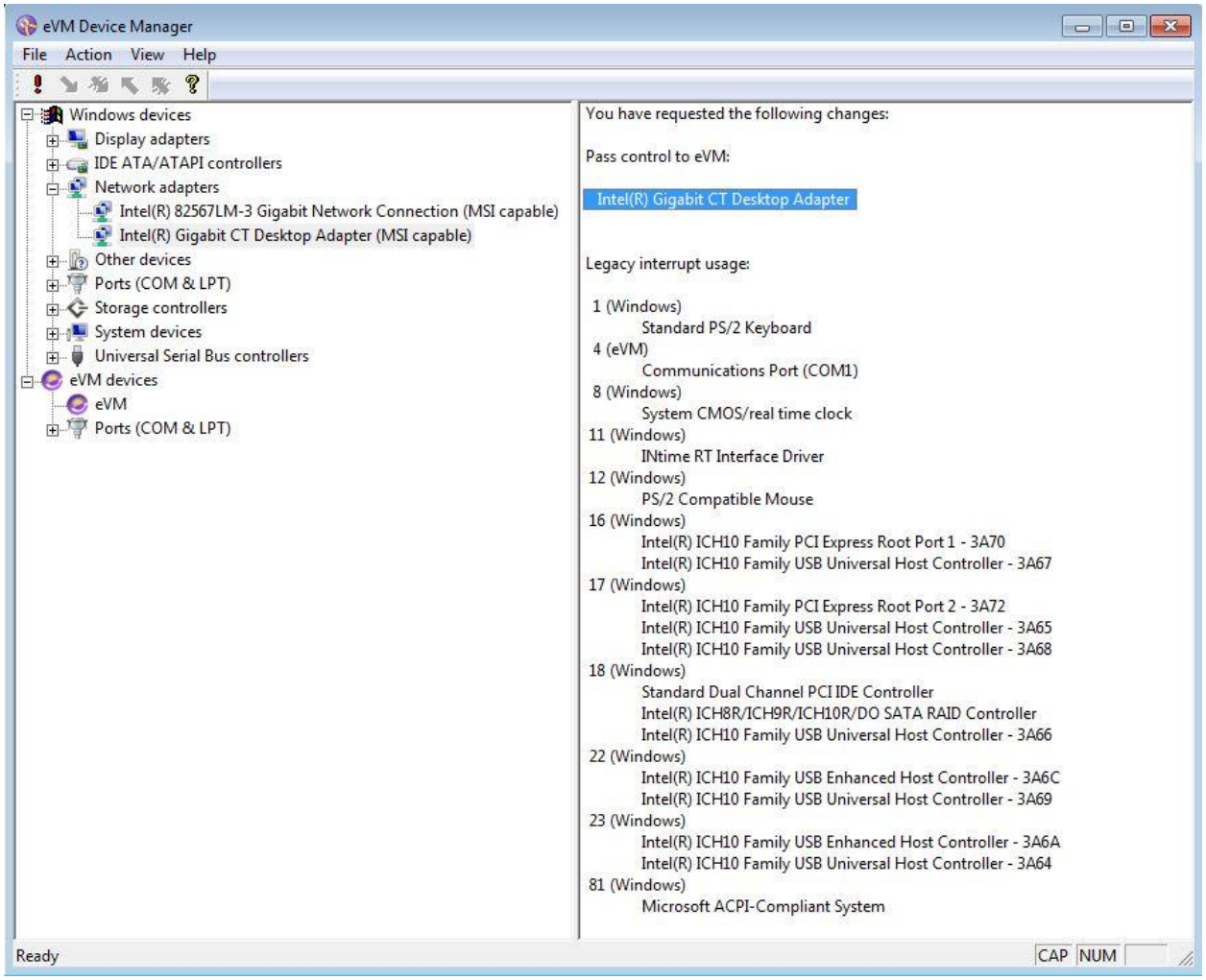

eVM Device Manager
File Action View Help
Windows devices
Display adapters
IDE ATA/ATAPI controllers
Network adapters
Intel(R) 82567LM-3 Gigabit Network Connection (MSI capable)
Intel(R) Gigabit CT Desktop Adapter (MSI capable)
Other devices
Ports (COM & LPT)
Storage controllers
System devices
Universal Serial Bus controllers
eVM devices
eVM
Ports (COM & LPT)
You have requested the following changes:
Pass control to eVM:
Intel(R) Gigabit CT Desktop Adapter
Legacy interrupt usage:
1 (Windows)
Standard PS/2 Keyboard
4 (eVM)
Communications Port (COM1)
8 (Windows)
System CMOS/real time clock
11 (Windows)
INtime RT Interface Driver
12 (Windows)
PS/2 Compatible Mouse
16 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 1 - 3A70
Intel(R) ICH10 Family USB Universal Host Controller - 3A67
17 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 2 - 3A72
Intel(R) ICH10 Family USB Universal Host Controller - 3A65
Intel(R) ICH10 Family USB Universal Host Controller - 3A68
18 (Windows)
Standard Dual Channel PCI IDE Controller
Intel(R) ICH8R/ICH9R/ICH10R/DO SATA RAID Controller
Intel(R) ICH10 Family USB Universal Host Controller - 3A66
22 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6C
Intel(R) ICH10 Family USB Universal Host Controller - 3A69
23 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6A
Intel(R) ICH10 Family USB Universal Host Controller - 3A64
81 (Windows)
Microsoft ACPI-Compliant System
Ready
CAP
NUM

Figure 4.55 - MSI Capable Device Pass Request

Save the device configuration, as before, and restart the platform; and the change will be applied to the platform. After saving the configuration and restarting, launch the eVM Device Manager and expand the **Network adapters** device nodes under *Windows devices* and *eVM devices*; and you will see that the network device has been removed from Windows control and is now dedicated to eVM for use by the guest OS, Figure 4.56.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

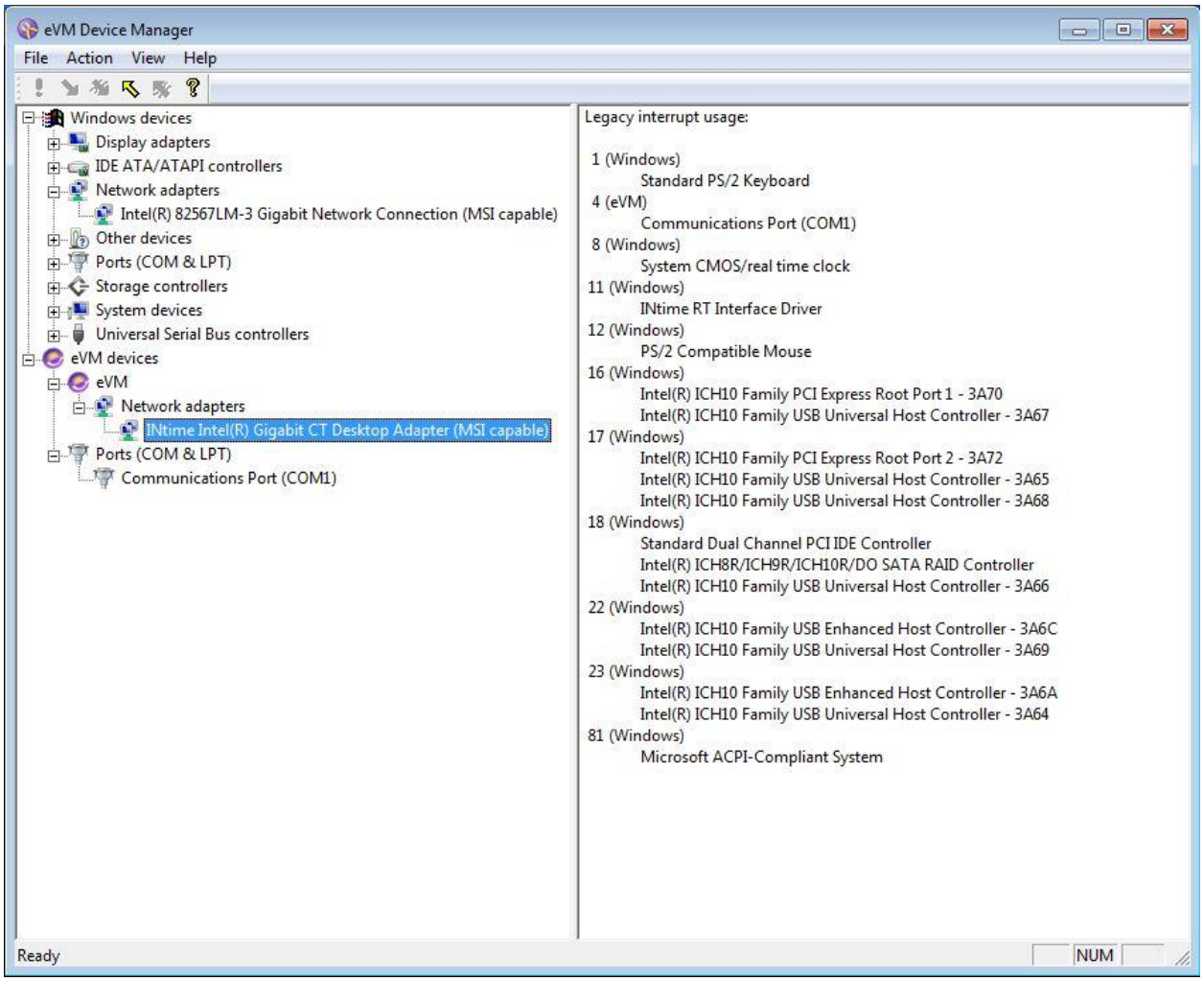

eVM Device Manager
File Action View Help
Windows devices
Display adapters
IDE ATA/ATAPI controllers
Network adapters
Intel(R) 82567LM-3 Gigabit Network Connection (MSI capable)
Other devices
Ports (COM & LPT)
Storage controllers
System devices
Universal Serial Bus controllers
eVM devices
eVM
Network adapters
INtime Intel(R) Gigabit CT Desktop Adapter (MSI capable)
Ports (COM & LPT)
Communications Port (COM1)
Legacy interrupt usage:
1 (Windows)
Standard PS/2 Keyboard
4 (eVM)
Communications Port (COM1)
8 (Windows)
System CMOS/real time clock
11 (Windows)
INtime RT Interface Driver
12 (Windows)
PS/2 Compatible Mouse
16 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 1 - 3A70
Intel(R) ICH10 Family USB Universal Host Controller - 3A67
17 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 2 - 3A72
Intel(R) ICH10 Family USB Universal Host Controller - 3A65
Intel(R) ICH10 Family USB Universal Host Controller - 3A68
18 (Windows)
Standard Dual Channel PCI IDE Controller
Intel(R) ICH8R/ICH9R/ICH10R/DO SATA RAID Controller
Intel(R) ICH10 Family USB Universal Host Controller - 3A66
22 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6C
Intel(R) ICH10 Family USB Universal Host Controller - 3A69
23 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6A
Intel(R) ICH10 Family USB Universal Host Controller - 3A64
81 (Windows)
Microsoft ACPI-Compliant System
Ready
NUM

Figure 4.56 - MSI Network Adapter Device Passed to eVM

4.6.5 Bus Master Warning Example

To illustrate the bus master warning that the eVM Device Manager will issue when passing an MSI device to a platform that does not support VT-d, we turned off VT-d support in the BIOS of our test platform, and check using VtProbe that VT-d support had been disabled, Figure 4.57.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

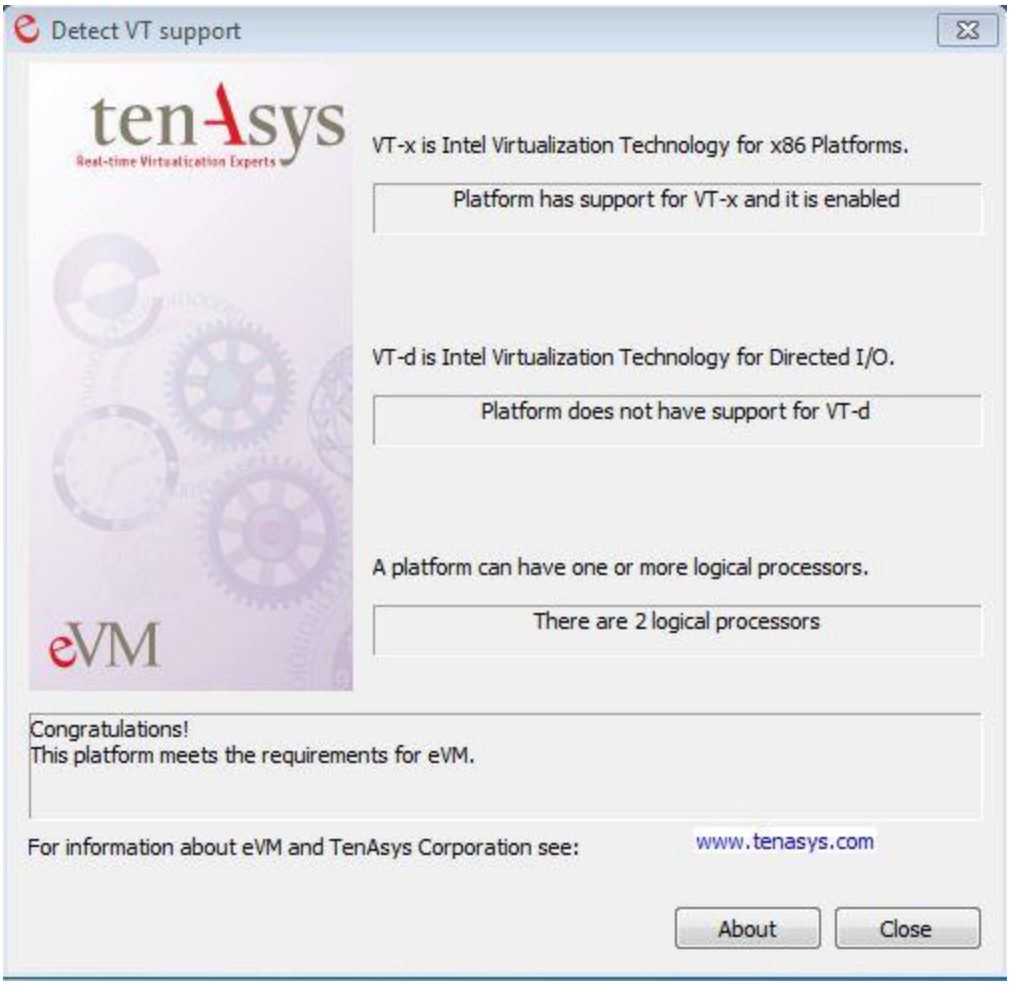

Detect VT support
tenAsys
Real-time Virtualization Experts
eVM
VT-x is Intel Virtualization Technology for x86 Platforms.
Platform has support for VT-x and it is enabled
VT-d is Intel Virtualization Technology for Directed I/O.
Platform does not have support for VT-d
A platform can have one or more logical processors.
There are 2 logical processors
Congratulations!
This platform meets the requirements for eVM.
For information about eVM and TenAsys Corporation see:
www.tenasys.com
About
Close

Figure 4.57 - VT-x Support Enabled but VT-d Support Disabled

Then we repeated the exercise in Section 4.6.4, passing the MSI network device to eVM. When we select **Pass to eVM**, the eVM Device Manager issues a warning for the network device, Figure 4.58.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

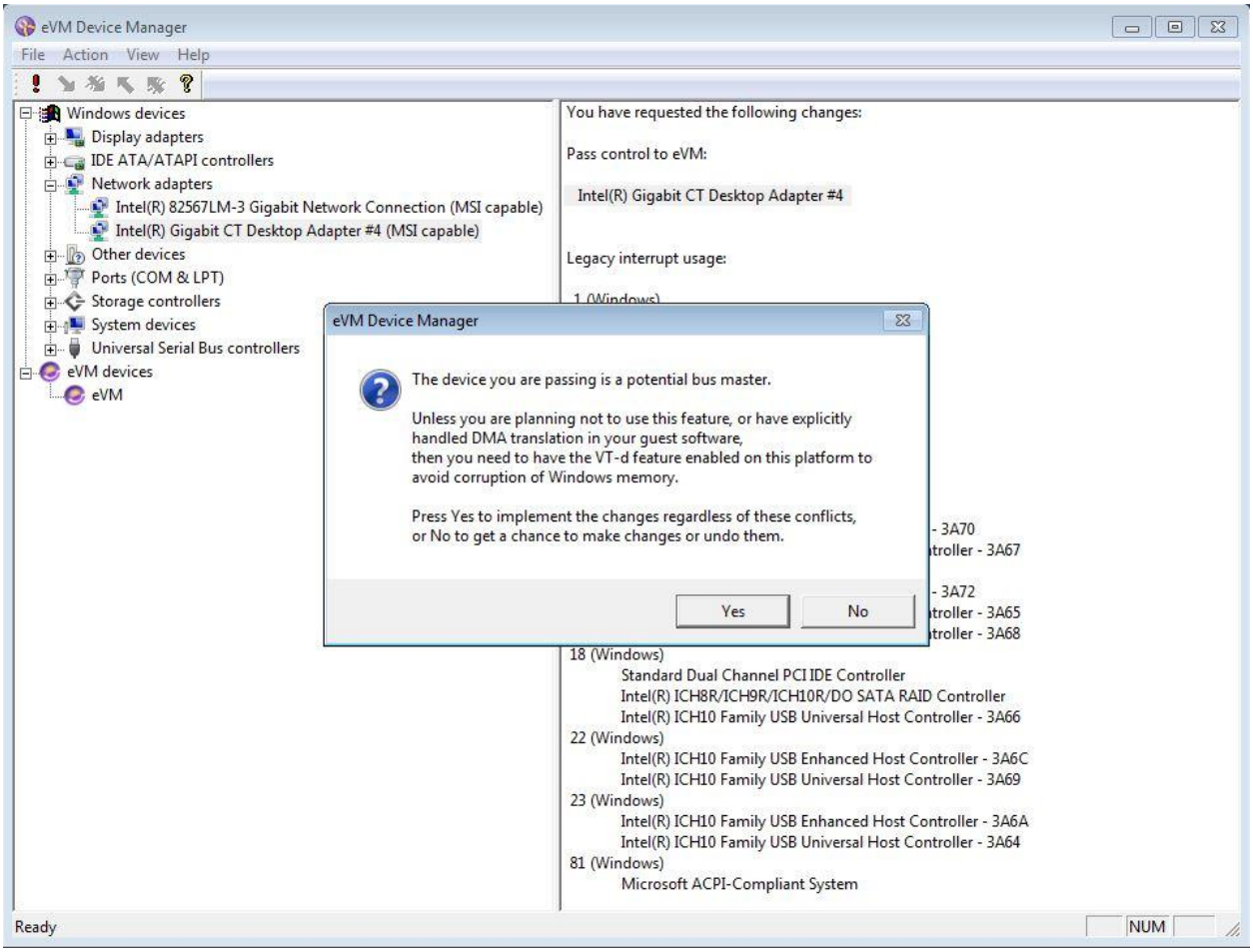

eVM Device Manager
File Action View Help
Windows devices
Display adapters
IDE ATA/ATAPI controllers
Network adapters
Intel(R) 82567LM-3 Gigabit Network Connection (MSI capable)
Intel(R) Gigabit CT Desktop Adapter #4 (MSI capable)
Other devices
Ports (COM & LPT)
Storage controllers
System devices
Universal Serial Bus controllers
eVM devices
eVM
You have requested the following changes:
Pass control to eVM:
Intel(R) Gigabit CT Desktop Adapter #4
Legacy interrupt usage:
eVM Device Manager
The device you are passing is a potential bus master.
Unless you are planning not to use this feature, or have explicitly handled DMA translation in your guest software, then you need to have the VT-d feature enabled on this platform to avoid corruption of Windows memory.
Press Yes to implement the changes regardless of these conflicts, or No to get a chance to make changes or undo them.
Yes
No
18 (Windows)
Standard Dual Channel PCI IDE Controller
Intel(R) ICH8R/ICH9R/ICH10R/DO SATA RAID Controller
Intel(R) ICH10 Family USB Universal Host Controller - 3A66
22 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6C
Intel(R) ICH10 Family USB Universal Host Controller - 3A69
23 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6A
Intel(R) ICH10 Family USB Universal Host Controller - 3A64
81 (Windows)
Microsoft ACPI-Compliant System
Ready
NUM

Figure 4.58 - Bus Master Warning for no VT-d Support

If you allow the eVM Device Manager to pass this device to eVM, such a condition would require that the bus master driver be modified to manage the address offset for the virtual machine, or bus master capability would have to be disabled in the device, if pOSesible.

4.6.6 Passing a Device back to Windows

Now, let's give the network adapter that was passed to eVM in Section 4.6.4, back to Windows. Select the **Intel® Gigabit CT Desktop Adapter** in the eVM *Network adapters* node in the left pane of the eVM Device Manager, and click on the **Pass to Windows** button, Figure 4.59.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

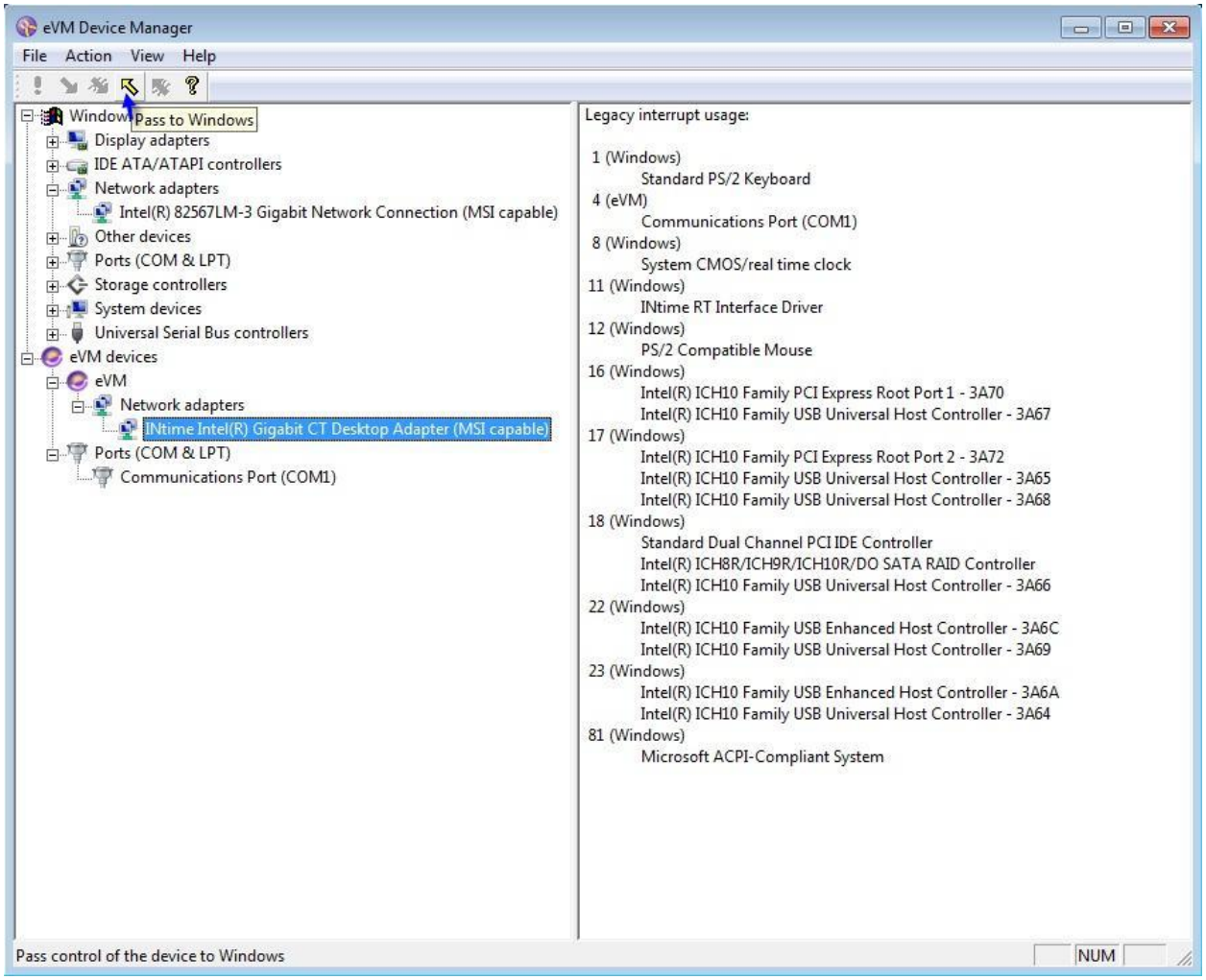

eVM Device Manager
File Action View Help
Pass to Windows
Windows
Display adapters
IDE ATA/ATAPI controllers
Network adapters
Intel(R) 82567LM-3 Gigabit Network Connection (MSI capable)
Other devices
Ports (COM & LPT)
Storage controllers
System devices
Universal Serial Bus controllers
eVM devices
eVM
Network adapters
INtime Intel(R) Gigabit CT Desktop Adapter (MSI capable)
Ports (COM & LPT)
Communications Port (COM1)
Legacy interrupt usage:
1 (Windows)
Standard PS/2 Keyboard
4 (eVM)
Communications Port (COM1)
8 (Windows)
System CMOS/real time clock
11 (Windows)
INtime RT Interface Driver
12 (Windows)
PS/2 Compatible Mouse
16 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 1 - 3A70
Intel(R) ICH10 Family USB Universal Host Controller - 3A67
17 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 2 - 3A72
Intel(R) ICH10 Family USB Universal Host Controller - 3A65
Intel(R) ICH10 Family USB Universal Host Controller - 3A68
18 (Windows)
Standard Dual Channel PCI IDE Controller
Intel(R) ICH8R/ICH9R/ICH10R/DO SATA RAID Controller
Intel(R) ICH10 Family USB Universal Host Controller - 3A66
22 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6C
Intel(R) ICH10 Family USB Universal Host Controller - 3A69
23 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6A
Intel(R) ICH10 Family USB Universal Host Controller - 3A64
81 (Windows)
Microsoft ACPI-Compliant System
Pass control of the device to Windows
NUM

Figure 4.59 - Select the Network Adapter and Pass It Back to Windows

Again, eVM Device Manger will show the request in the right pane, except this time it shows that we are requesting to pass the device to Windows, Figure 4.60.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

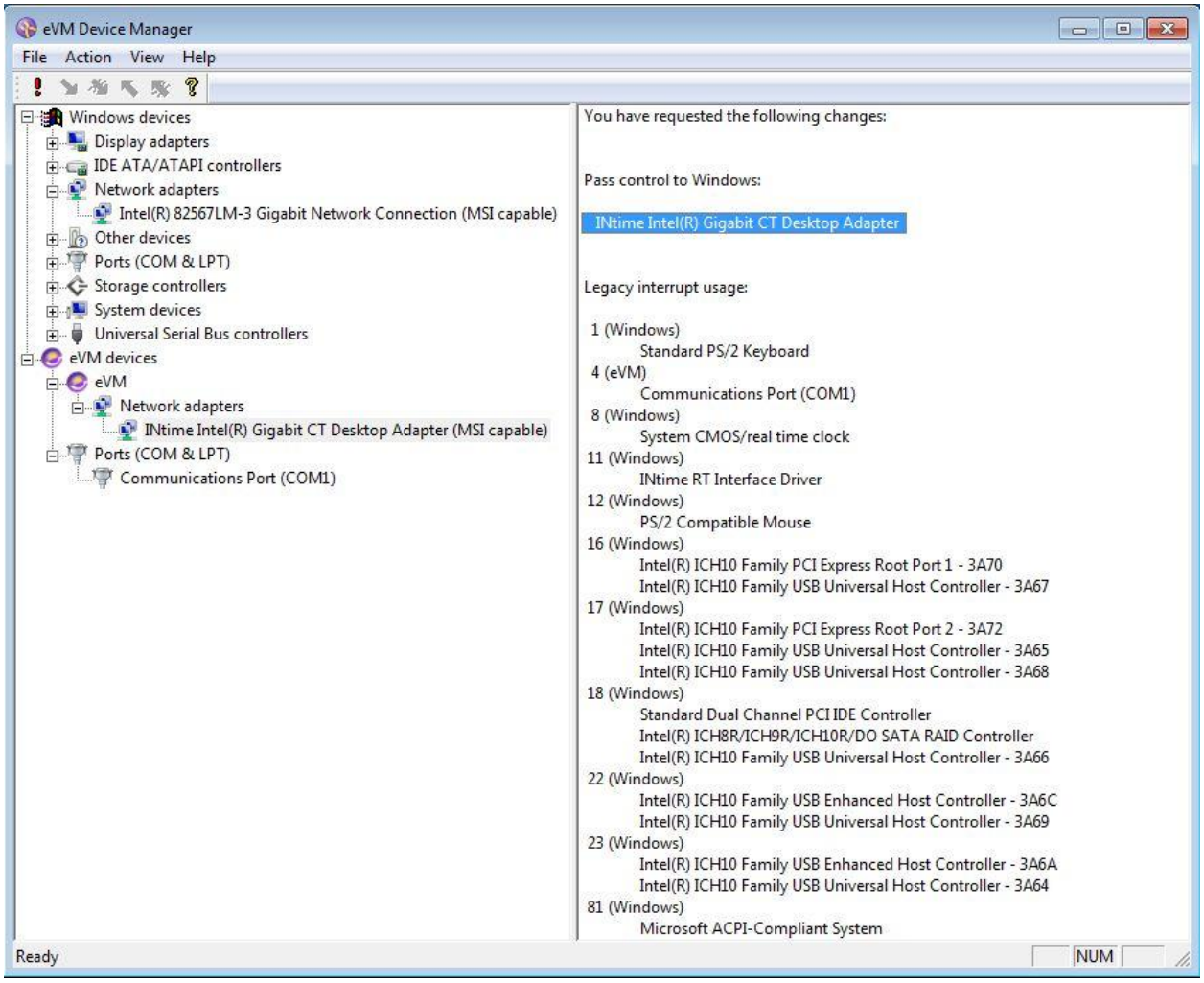

eVM Device Manager
File Action View Help
Windows devices
Display adapters
IDE ATA/ATAPI controllers
Network adapters
Intel(R) 82567LM-3 Gigabit Network Connection (MSI capable)
Other devices
Ports (COM & LPT)
Storage controllers
System devices
Universal Serial Bus controllers
eVM devices
eVM
Network adapters
INtime Intel(R) Gigabit CT Desktop Adapter (MSI capable)
Ports (COM & LPT)
Communications Port (COM1)
You have requested the following changes:
Pass control to Windows:
INtime Intel(R) Gigabit CT Desktop Adapter
Legacy interrupt usage:
1 (Windows)
Standard PS/2 Keyboard
4 (eVM)
Communications Port (COM1)
8 (Windows)
System CMOS/real time clock
11 (Windows)
INtime RT Interface Driver
12 (Windows)
PS/2 Compatible Mouse
16 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 1 - 3A70
Intel(R) ICH10 Family USB Universal Host Controller - 3A67
17 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 2 - 3A72
Intel(R) ICH10 Family USB Universal Host Controller - 3A65
Intel(R) ICH10 Family USB Universal Host Controller - 3A68
18 (Windows)
Standard Dual Channel PCI IDE Controller
Intel(R) ICH8R/ICH9R/ICH10R/DO SATA RAID Controller
Intel(R) ICH10 Family USB Universal Host Controller - 3A66
22 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6C
Intel(R) ICH10 Family USB Universal Host Controller - 3A69
23 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6A
Intel(R) ICH10 Family USB Universal Host Controller - 3A64
81 (Windows)
Microsoft ACPI-Compliant System
Ready
NUM

Figure 4.60 - Request to Pass the Network Controller back to Windows

The eVM Device Manager will prompt you to restart the platform to apply this change, Figure 4.61.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

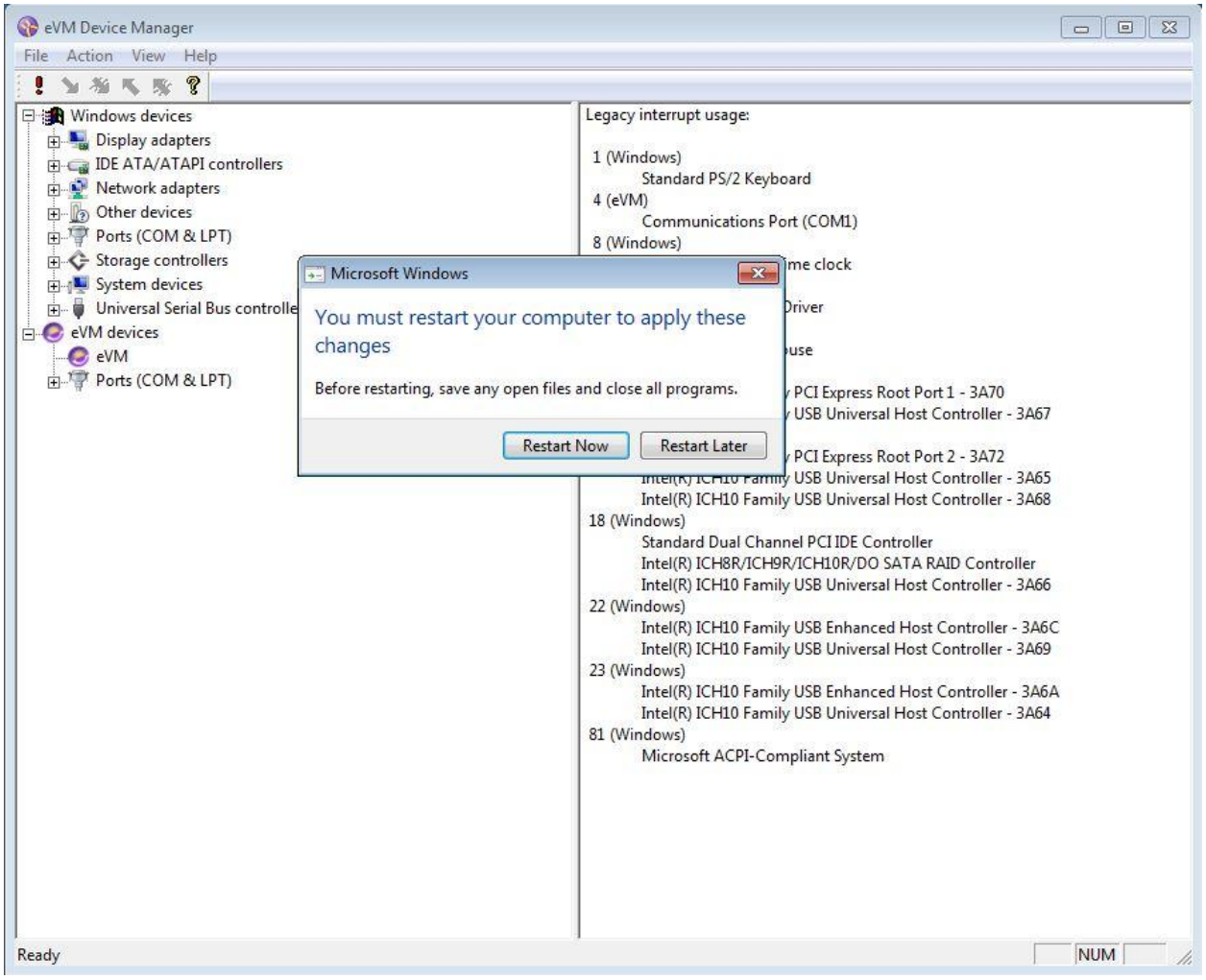

eVM Device Manager
File Action View Help
Windows devices
Display adapters
IDE ATA/ATAPI controllers
Network adapters
Other devices
Ports (COM & LPT)
Storage controllers
System devices
Universal Serial Bus controlle
eVM devices
eVM
Ports (COM & LPT)
Legacy interrupt usage:
1 (Windows)
Standard PS/2 Keyboard
4 (eVM)
Communications Port (COM1)
8 (Windows)
Microsoft Windows
You must restart your computer to apply these changes
Before restarting, save any open files and close all programs.
Restart Now
Restart Later
PCI Express Root Port 1 - 3A70
USB Universal Host Controller - 3A67
PCI Express Root Port 2 - 3A72
Intel(R) ICH10 Family USB Universal Host Controller - 3A65
Intel(R) ICH10 Family USB Universal Host Controller - 3A68
18 (Windows)
Standard Dual Channel PCI IDE Controller
Intel(R) ICH8R/ICH9R/ICH10R/DO SATA RAID Controller
Intel(R) ICH10 Family USB Universal Host Controller - 3A66
22 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6C
Intel(R) ICH10 Family USB Universal Host Controller - 3A69
23 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6A
Intel(R) ICH10 Family USB Universal Host Controller - 3A64
81 (Windows)
Microsoft ACPI-Compliant System
Ready
NUM

Figure 4.61 - Restart Request to Apply Device Configuration Changes

After restarting the platform, launching the eVM Device Manager, and expanding the device nodes in the left pane, it can be seen that the network adapter is back under Windows control, Figure 4.62.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

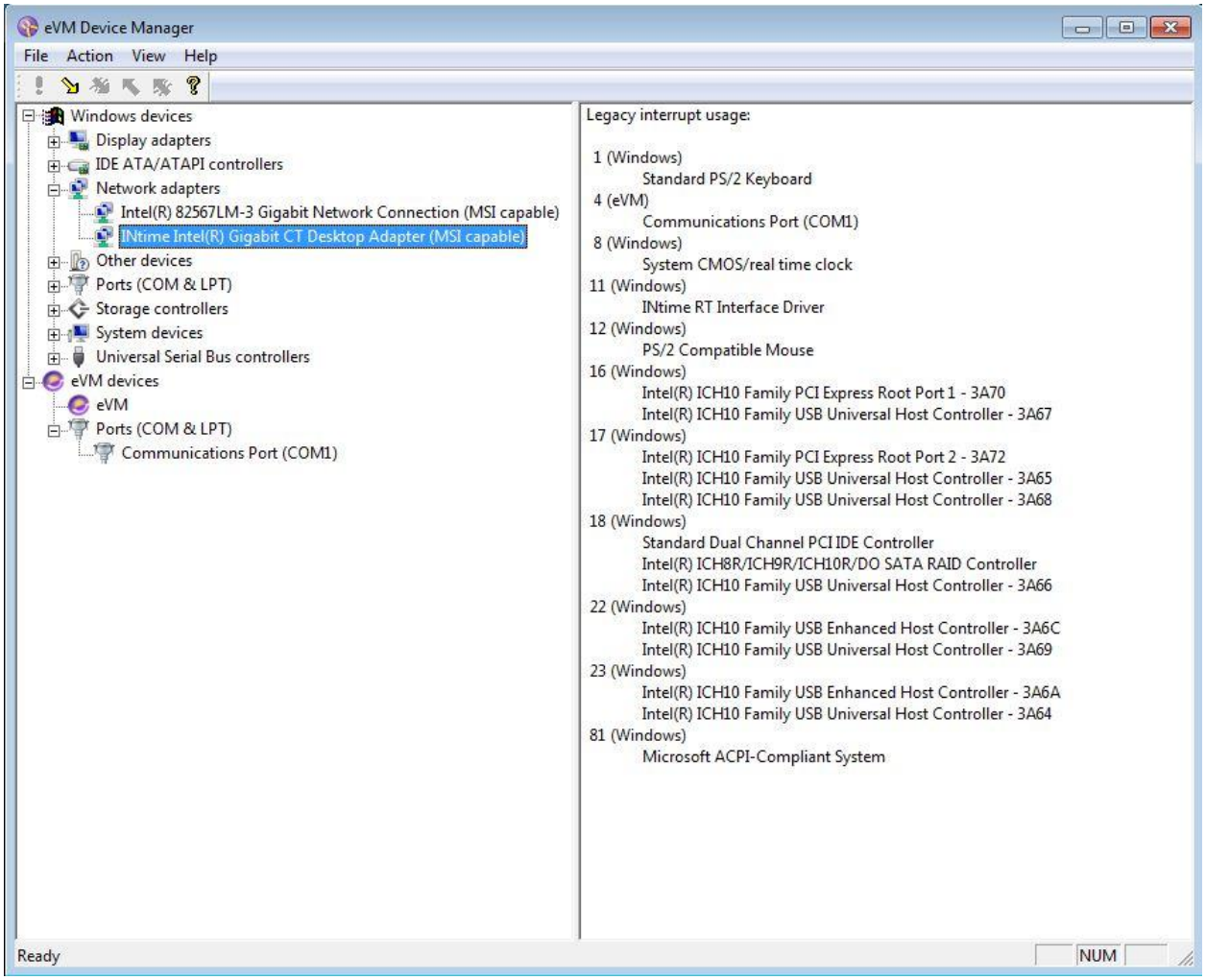

eVM Device Manager
File Action View Help
Windows devices
Display adapters
IDE ATA/ATAPI controllers
Network adapters
Intel(R) 82567LM-3 Gigabit Network Connection (MSI capable)
INtime Intel(R) Gigabit CT Desktop Adapter (MSI capable)
Other devices
Ports (COM & LPT)
Storage controllers
System devices
Universal Serial Bus controllers
eVM devices
eVM
Ports (COM & LPT)
Communications Port (COM1)
Legacy interrupt usage:
1 (Windows)
Standard PS/2 Keyboard
4 (eVM)
Communications Port (COM1)
8 (Windows)
System CMOS/real time clock
11 (Windows)
INtime RT Interface Driver
12 (Windows)
PS/2 Compatible Mouse
16 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 1 - 3A70
Intel(R) ICH10 Family USB Universal Host Controller - 3A67
17 (Windows)
Intel(R) ICH10 Family PCI Express Root Port 2 - 3A72
Intel(R) ICH10 Family USB Universal Host Controller - 3A65
Intel(R) ICH10 Family USB Universal Host Controller - 3A68
18 (Windows)
Standard Dual Channel PCI IDE Controller
Intel(R) ICH8R/ICH9R/ICH10R/DO SATA RAID Controller
Intel(R) ICH10 Family USB Universal Host Controller - 3A66
22 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6C
Intel(R) ICH10 Family USB Universal Host Controller - 3A69
23 (Windows)
Intel(R) ICH10 Family USB Enhanced Host Controller - 3A6A
Intel(R) ICH10 Family USB Universal Host Controller - 3A64
81 (Windows)
Microsoft ACPI-Compliant System
Ready
NUM

Figure 4.62 - Network Adapter Back Under Windows Control

4.7 Shared Memory Interface

4.7.1 Description

Shared memory for passing data from a guest OS process and Windows processes is allocated from the Windows non-paged pool as a single block of memory and the assigned to eVM for management. Part or all of the shared memory may be mapped to a Windows process while the guest application access it via a virtual PCI device which exposes the memory through a BAR (Base Address Register) in the PCI configuration header for the device.

A synchronization mechanism is provided to provide a "doorbell" functionality that allows one side to signal the other when memory is changed. This signal is delivered to a Windows application via an API, and to a guest application via an interrupt from the virtual PCI device.

4.7.2 Configuration

Size of the shared memory is setup with the *Node Management* in the *eVM Configuration Panel*. Select the *System* tab and go down to the *Shared Memory size* entry. Valid sizes include any number of 0 to 256 bytes in 4 bytes increments. See Figure 4.63.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

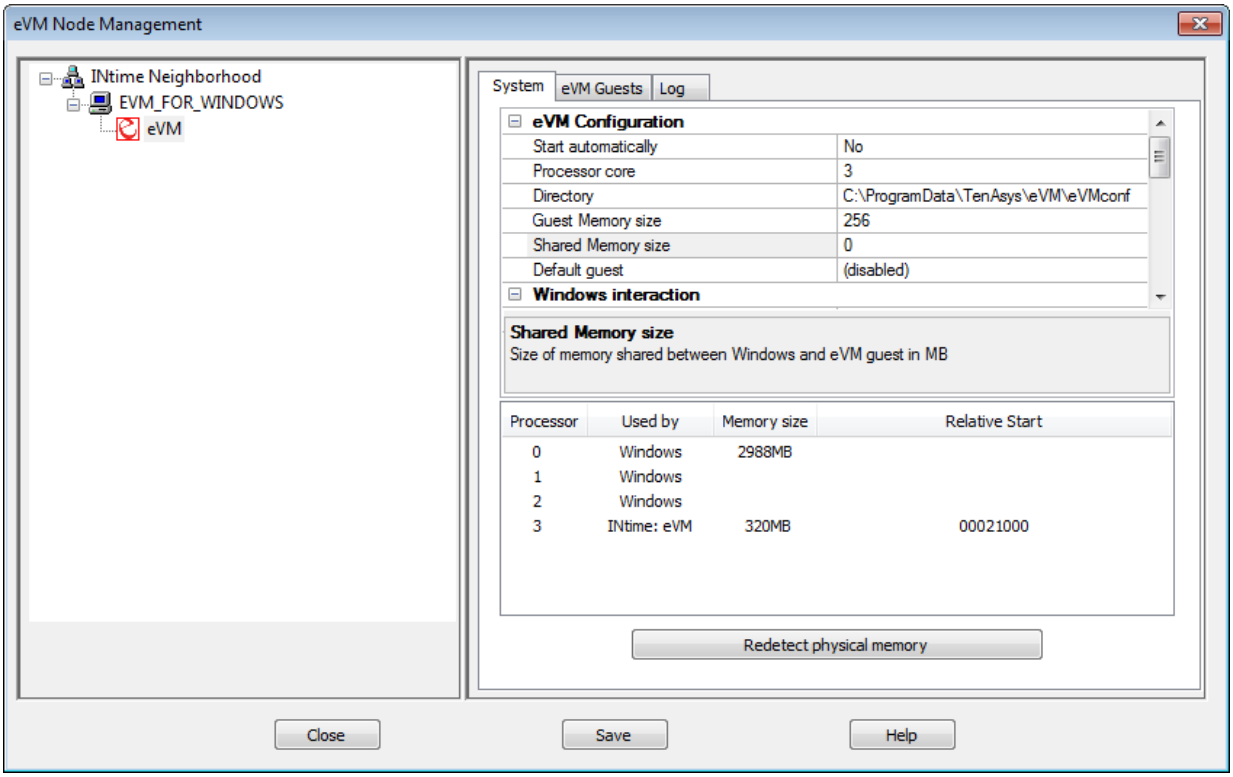


Figure 4.63 Setting size of *Shared Memory*

IRQ for guest access is set in the *eVM Guests* tab of the *eVM Node Management* window. Go to the *Shared Memory* line and select ***yes*** next to the *Enable shared memory* line. This will add another line listing *Memory write interrupt* and an interrupt (number) that has been allocated to it. See Figure 4.64.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

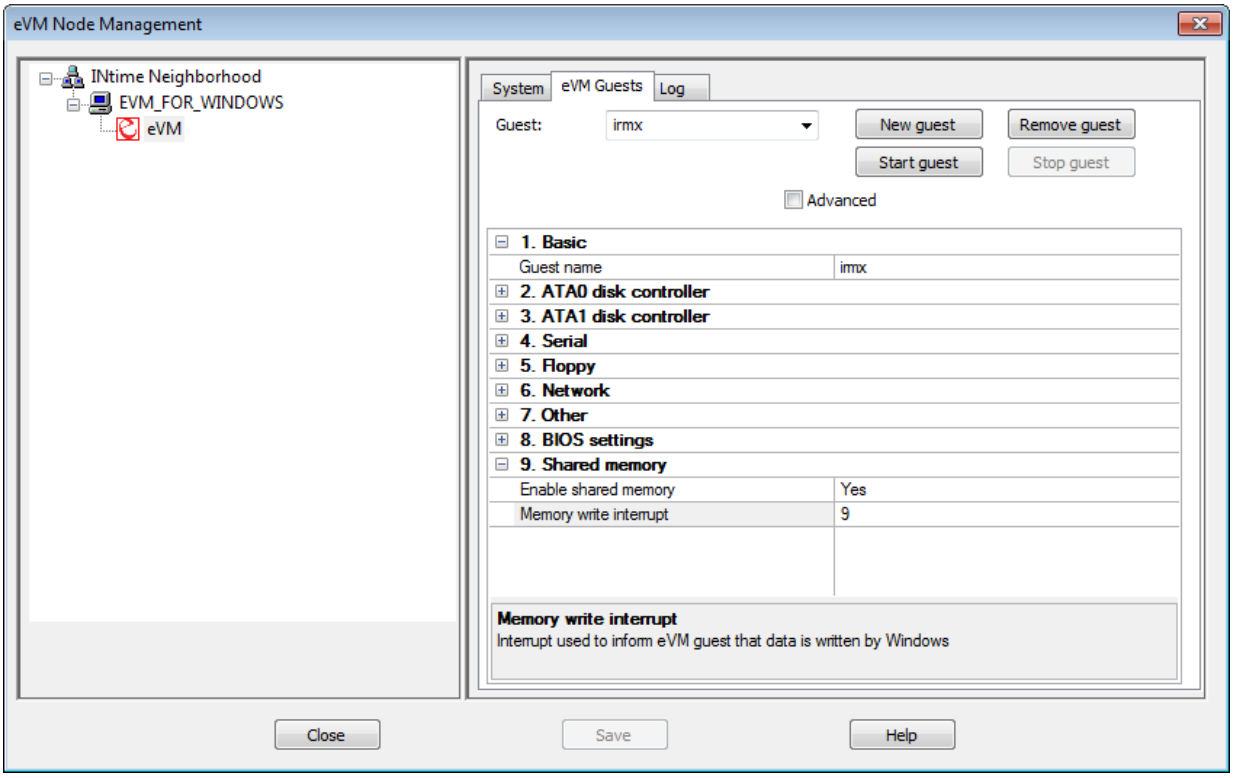


Figure 4.64 Setting Shared Memory IRQ for guest application.

4.7.3 Shared Memory Usage

4.7.3.1 Usage from the Windows application side

To access the shared memory buffer the Windows application must make use of the API described in Appendix F. First, the application must first determine the configured size of the shared memory by calling *evmGetSharedMemoryInfo*, then mapping all or part of it by calling *evmMapSharedMemory*.

Call *evmWaitForSharedMemory* to wait for a signal from the guest application.

Call *evmSignalSharedMemory* to send a signal to the guest operating system.

Call *evmUnmapSharedMemory to* unmap the shared memory from the application.

4.7.3.2 Usage from the guest application side

Guest application gains access to the configured shared memory area by means of a virtual PCI device. First the application must discover the device with PCI Vendor ID 0x10b5 and Device ID 0x9050. If required an interrupt handler may be set up for the device using the interrupt line data found in the IntLine register in the PCI header. The shared memory is available via BAR4 in the device header, and the control and status registers are available via BAR3.

When the Windows application calls *evmSignalSharedMemory*, an interrupt is generated from this device. When a value is written to the device's control register, any Windows thread which has called e*vmWaitForSharedMemory* will be woken up.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

4.7.4 PCI configuration space definition

The virtual PCI device has the following minimal PCI configuration space definition:

Offset	Size (bits)	Name	Default value	Comments
0x00	16	VENDOR_ID	0x10b5	RO
0x02	16	DEVICE_ID	0x9050	RO
0x04	16	COMMAND	0	Bit 0, 1 and 8 only are implemented.
0x06	16	STATUS	0x0280	RO
0x08	8	REVISION_ID	0x01	RO
0x09	24	CLASS_CODE	0x068000	“Other bridge device”
0x0e	8	HEADER_TYPE	0x00	Single-function PCI device
0x1c	32	BAR3	0x00000001	Register space I/O address
0x20	32	BAR4	0x00000000	Shared memory physical address
0x2c	16	SUBSYSTEM_VENDOR_ID	0x0000	RO
0x2e	16	SUBSYSTEM_DEVICE_ID	0x0002	RO
0x34	8	CAPABILITIES_PTR	0x00	No capabilities
0x3c	8	INTERRUPT_LINE	0x00	Programmed if the interrupt is enabled by configuration
0x3d	8	INTERRUPT_PIN	0x01	RO

4.7.5 Register definition

The following registers are defined within the I/O space described by BAR3.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

4.7.5.1 Shared memory command register – offset 0 in BAR3

This register is a R/W, 32-bit register which is used to configure and use the device. Command words are written to the register to perform specific actions. Valid command values are as follows:

Value	Name	Description
0x00000001	ENABLE_INTR	Enables the device to generate interrupts. An interrupt will be generated when a doorbell signal is sent to the device from Windows.
0x00000002	DISABLE_INTR	Disables interrupt-generation.
0x00000003	RING_DOORBELL	Sends a doorbell signal to Windows. Any thread waiting in evmWaitForSharedMemory will be awakened. The state is sticky; if no thread was waiting, the next thread to call evmWaitForSharedMemory will return immediately.
0x00000004	RESET_DOORBELL	Cancels any outstanding doorbell signals to Windows.

4.7.5.2 Shared memory status register – offset 4 in BAR3

This register is a read-only register. Writing to it has no effect.

31-5	4	3-2	1	0
(unused)	ASSERTED	(unused)	INTR_PEND	DOORBELL

When a doorbell signal is sent to Windows , the ASSERTED bit will be set until a Windows application received the signal.
When a doorbell signal is sent from Windows, the DOORBELL bit will be set, and if interrupts are enabled, the INTR_PEND bit will be set until the next read of the status register.

4.7.6 Typical operation

A typical sequence of operations by the guest software is as follows:

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

1. Discover the device using normal PCI methods.
2. Enable I/O and memory access by writing 0x0003 to the PCI COMMAND register.
3. Using the value in the INT_LINE PCI register, install an interrupt handler.
4. Map the shared memory using the physical address in BAR4
5. Write ENABLE_INTR to the command register to enable interrupt on the device.

Now the memory may be written and read, as it can from the Windows application. Synchronization of access to the memory may be performed using the doorbell mechanism, or spinlocks may be created in the shared memory for access from both sides.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

A. Production installation

After you have set up eVM to meet your application requirements on your target development system, you are probably going to want to replicate that setup in a production environment. To do that without having to configure each and every installation of eVM in production, you can save the configuration from the target development system that is already set up and import that saved configuration to each of the production systems.

This operation is done with the ***evmconf.exe*** utility that has been loaded as part of the eVM for Windows installation. This utility can be executed from the Windows ***Run*** command by pressing and holding the ***Windows*** key and then hitting the '***R***' key. In the Run dialog you can enter the command line parameters described below, or you can also launch the ***CMD*** Shell from the Windows ***Run*** command and enter the command line parameters described below at the command prompt. In Windows Vista or Windows 7 you can also enter the command line parameters described below in Windows ***Start Search***.

Note: for the "/n" command line switches, 'n' is case insensitive.

A.1 Saving the configuration with the /e: option

Syntax usage:
evmconf /e:[export file name and full path]

Once you have configured eVM on your target development system, the /e option of ***evmconf.exe*** will allow you to export the known working configuration to a file that can be used later to import into other systems to configure them to the same setup parameters as the target development system that this configuration was exported from, or it can also be used to restore the eVM configuration of your target development system to this known state, if you have made configuration changes after saving this configuration.

In the example below we have a system that is already configured whose configuration has been labeled "My Old OS"

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

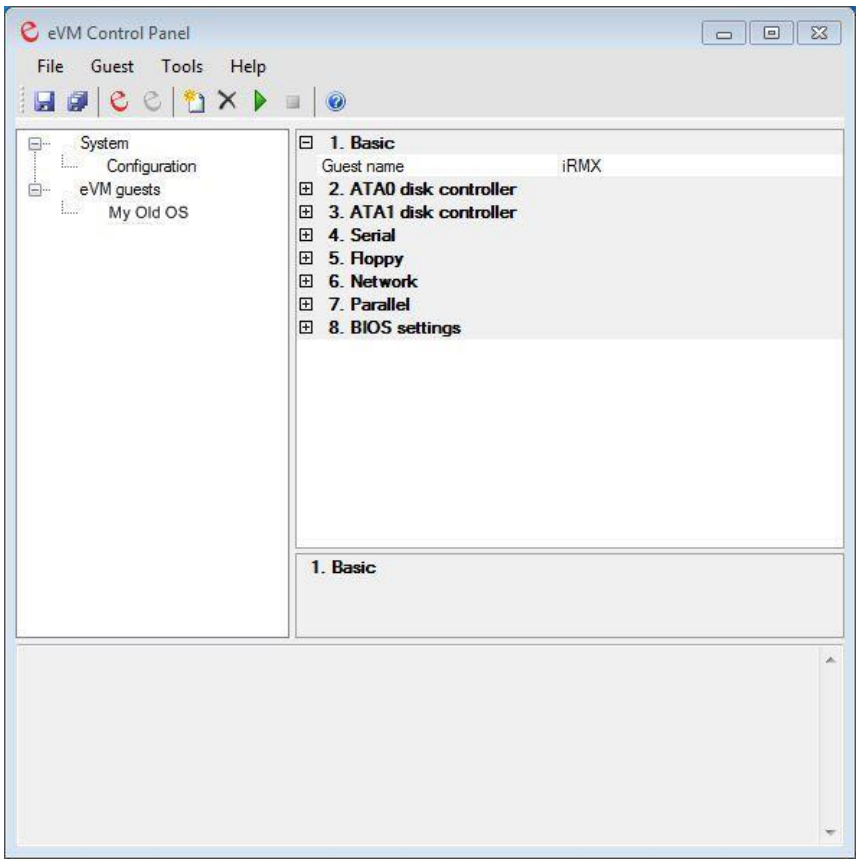


Figure 4.65 - eVM Configuration for My Old OS

We are going to save that configuration in a file named MyeVMConf.ecf by, first closing the eVM Control Panel and then entering the command shown below from the ***run, CMD*** Shell***,*** or ***Start Search*** command line.

evmconf /e:c:\MyeVMConf.ecf
This will create a file called ***MYeVMConf.ecf*** in the root directory of the C: drive.
Note: any file extension can be used. The ".ecf" extension is the default.

A.2 Restoring or importing a known configuration

When you first install eVM, there are no settings included in the configuration, as shown in the screen capture Figure 4.66 below. The ***/i*** option of the ***evmconf.exe*** command allows you to import a known setting from another system and allows you to restore a saved setting to a modified configuration.

Syntax usage:
evmconf /i:[saved configuration file name and full path]

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

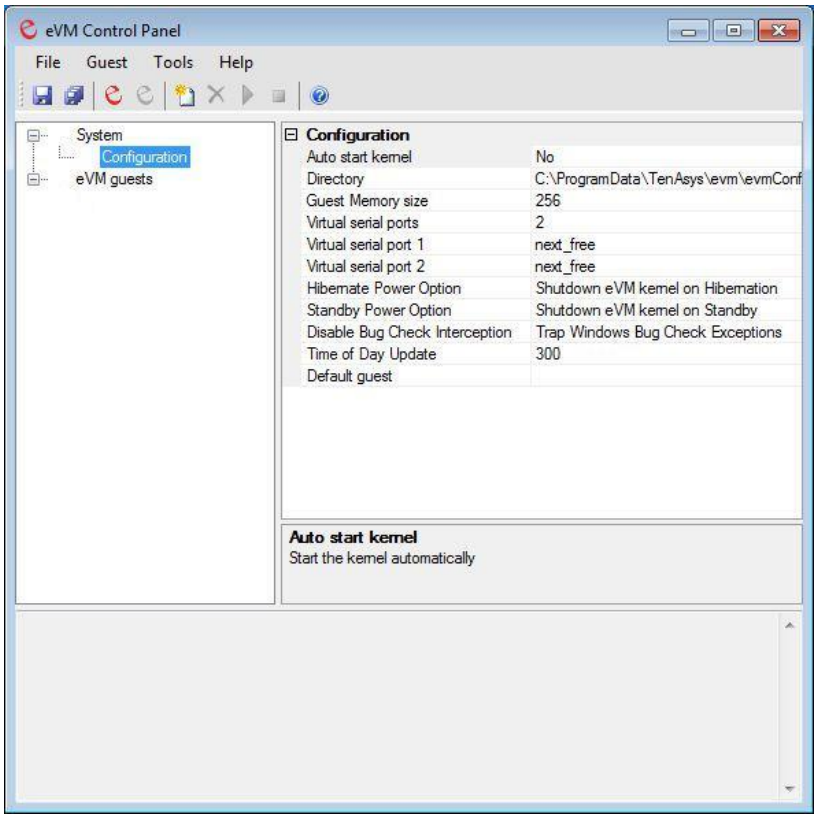


Figure 4.66 - eVM Configuration

To import the settings from the target development system to a new production system, first close the eVM Control Panel.

Second, copy the configuration ***MyeVMConf.ecf*** file to the root directory of the C: drive of the of each new production system.

Then import the configuration by entering the command shown below from the ***run***, ***CMD*** Shell***,*** or ***Start Search*** command line.

evmconf /i:c:\MyeVMConf.ecf

Opening the EVM Control Panel on the production system should display a configuration labeled ***My old OS*** with the settings of the imported configuration. If this was used to restore the configuration of a modified system, the EVM Control Panel should show that the modified system settings have been restored to the ***My Old OS*** settings.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

A.3 Assigning devices to eVM on the command line

PCI devices

When the guest OS needs full control of a PCI device, the PCI device should be passed to the guest. This can be done from within the eVM Device Configuration utility, or it can be done from the command line:

DevConfig /option "device" [/instance n]

Where '*option*' is either of:

Toevm - pass a device to eVM and let the guest use the interrupt as defined in the current configuration. The interrupt should not be shared with another device that is controlled by Windows.

Toevmnoirq - pass a device to eVM and let the device use MSI (Message Signaled Interrupts); a unique interrupt number will be assigned.

"device" - must be the full device name as shown in the eVM Device Configuration utility under the eVM tree. You may replace trailing characters by a single star (*), if that still leaves a unique device name.

When more than one device matches the full device name, an instance number must be specified: zero indicates the first instance.

A.4 Command Line Installation of Virtual Serial Ports

When the guest OS needs one or more virtual serial ports, these can also be installed from within the eVM Control Panel or from the command line:

evmconf /s:[number]

Where '*number* 'is the number of virtual serial ports you want to install (from **0** to **4**).

The configuration result is the same as opening the eVM Control Panel; and in the System Configuration properties, setting Virtual Serial Ports to '*number*', setting each virtual serial port to 'next_free', and then saving the system configuration. Using '0' will remove all virtual serial ports.

A.5 Debug logging

When eVMconf is running, debug information is presented in the bottom pane of the window. You can choose to save the session's logging to a file, evmconf.log, in the %EVMCONF%

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

directory tree. Please note that the file will only save the current session, and a new session will start with a new blank file.

evmconf /l

You can also log additional information to the file with the /v, verbose, switch

evmconf /l /v

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

B. eVM Virtual Machine

The TenAsys eVM product presents a virtual machine to guest software within a Windows environment which emulates an Intel x86 based PC with certain chipset and other characteristics. In order to run eVM there is at least an Intel dual-core platform required that supports VT-x. The specification for the virtual machine created by eVM is described as follows.

B.1 System Requirements

The minimum requirement to run eVM is a PC platform running Windows XP, Windows Vista or Windows 7 with an Intel dual-core processor or single-core processor with hyperthreading and VT-x supported by the CPU and enabled in the BIOS.

B.2 Base Virtual Machine Specification

The base platform consists of the following components:

CPU

One core of the platform processor is used for the virtual machine. The other core or cores are used by Windows. Most of the native characteristics are available but some features are hidden, such as support for VT-x (VT-x features are not supported in the guest operating system). CPUID instructions are intercepted and some return values modified to hide features. The registers modified are the four returns by CPUID with EAX=1 and all bits are configurable internally. By default the following bits are disabled (always return 0) in ECX: VMX_CAPABLE (bit 5). The following are disabled by default in EDX: HTT, MTRR, APIC, PAE

Memory

Memory is allocated as a single physically-contiguous region to the guest software, and the VMM takes care of translating the physical addresses generated by the guest software.

Paging is emulated using the VTLB algorithm defined by Intel in this version of the VMM. Systems which do not perform well under the VTLB algorithm are demand-paging systems which frequently reload the page directory pointer (such as Windows or Linux). As VT-x2 hardware features become available this implementation will be superseded, resulting in much improved performance for such guest operating systems.

Hardware generated memory addresses (from a PCI bus master device for example) will be automatically translated if the base platform supports and enables VT-d technology; otherwise a change will need to be made to any software, such as device drivers, which calculate a physical address. An interface is provided to the guest software by the VMM which generates the requested offset when making such a calculation.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

Interrupt controller (8259A PIC)

The standard interrupt controller is the twin 8259-PIC architecture found in the legacy PC platform. This is emulated in the VM. All modes are supported except non-8086 mode and Special Fully Nested Mode.

The default interrupt map for the VM is as follows:

IRQ	Usage
0	System timer interrupt (from 8254)
1	(reserved)
3	Serial port 1 (COM2)
4	Serial port 0 (COM1)
5	PCI IRQ or
6	Floppy disk controller, or PCI IRQ
7	Spurious interrupt, or PCI IRQ
8	CMOS/RTC
9	PCI IRQ
10	PCI IRQ
11	PCI IRQ
12	NE2000
13	Legacy FPU error
14	Primary IDE controller
15	Secondary IDE controller

Table B 1 Virtual IRQ's

APIC mode will also be supported in an early update of the VMM.

System Timer (8254 PIT)

The system timer is the standard 8254 Programmable Interval Timer found in the legacy PC platform. It is emulated in the VM. Modes 1, 4 and 5 are currently not supported, nor is any BCD counting mode or 8-bit counting modes. Port 61 bits 1 and 4 are functional (speaker data and refresh timer bits).

CMOS & Real Time Clock

The CMOS access ports 70 and 71 are fully supported, and a CMOS RAM image is built at guest boot time from the eVM configuration data. The real-time clock is accessible through the CMOS ports in the normal way, and it will generate IRQ 8 as expected.

Port 92

Port 92 bits 0 (“high-speed reset”) and 1 (“A20 control”) are supported. Other port 92 bits are not currently supported.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

Serial ports (16550 UART)

An emulation of the standard 16550 UART is supported by the VMM and up to four serial ports are configurable. All modes of the 16550 UART are supported. Accesses to the virtual device may be either passed through to a real UART or else the virtual UART transmits and receives data to and from a Windows virtual serial driver, so a virtual serial link may be established between Windows and the guest software.

DMA controller (8237 DMA controller)

The legacy 8237 DMA controller pair is virtualized by the VMM. Currently the only device supported by the virtual controller is the virtual floppy disk controller.

Floppy Disk Controller

This device simulates a standard floppy disk controller. Two devices are supported (drives A: and B:) and only 1.44 Mbyte diskettes are currently virtualized. The media itself is a disk image of floppy disk medium, served by a Windows process to the VMM.

IDE controller

The standard IDE controller is virtualized by the VMM and presents two IDE controllers (primary and secondary) to the VM. Each virtual disk is actually a device image presented by the VMM from a Windows process. Virtual hard disk drives and CDROM devices are supported. The device image takes the form of a binary file. There will also be support for access to whole devices such as disks, or FLASH memory devices, in an early update of the VMM.

PCI Host Bridge (440FX Host Bridge, PCI device 0)

A virtualized 440FX host bridge (vendor ID 0x8086, device ID 0x1237) is presented by the VMM, which allows it to present a virtualized PCI local bus. PCI configuration space accesses are intercepted by the virtual host bridge which allows other virtual and real PCI devices to be inserted into the virtual PCI local bus.

ISA Bridge (440FX ISA Bridge, PCI ID 8086/7000)

This virtual device is primarily used to route PCI interrupts to the interrupt controller, and to emulate the APM registers (mostly non-functional since power management is not emulated in the VM).

The VM provides four PCI IRQ links (PIRQA through PIRQD) which are routed via this device to inputs on the interrupt controller. The routing is initialized according to the VM configuration at startup, and dynamically controlled through the IRQ routing registers in this device.

EIDE Controller (PCI ID 8086/7010)

This device emulates the EIDE controller, providing EIDE functionality for the disk controller. The DMA functionality is provided by the VMM software.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

ACPI Controller (PIIX4 ACPI Controller, PCI ID 8086/7113)

This device emulates the ACPI hardware controller for the chipset. Power management features are not emulated in the VM but all other ACPI hardware features are enabled.

B.3 Optional Virtual Machine Components

Ethernet controller

An NE2000 virtual controller is provided which provides a shared-memory and non-interrupt driven Ethernet link to a Windows Ethernet driver.

Shared memory device

A shared memory device will be provided which allows for the development of a simple interface to a memory area shared between Windows and the guest operating system. A simple signaling mechanism is provided by the interface which generates interrupts from the virtual device. The Windows side of the interface is handled through an API provided in the product.

B.4 BIOS

The VM has a BIOS which provides the standard functionality of a PC BIOS, including all of the functions required to boot a guest operating system. The default action of the eVM is to execute the BIOS when first started, which then attempts to boot the guest operating system via the virtual IDE or floppy interface from a disk or CDROM image.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

C. Guest OSes Currently Enabled:

The following OSes have been tested in basic configurations and the intention is to provide an adaptable configuration of the targeted guest OS with eVM. These need to support headless operation with a serial console mode.

- Microsoft Win CE 6.0
- QNX Neutrino
- WindRiver VxWorks
- SuSE Linux
- Siemens RMOS
- TenAsys iRMX

For an up-to-date listing of enabled guest OSes, check the following website: www.tenasys.com, and click on **Support->TenAsys Knowledge Base**.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

D. VtProbe

VtProbe is one of the tools contained in the eVM installation. It is used during installation to determine if the hardware platform has the required virtualization features to support eVM. It can also be run standalone to manually check a hardware platform for eVM compatibility.

To run VtProbe, do a **Start->Run** and enter **vtprobe** or enter **vtprobe** in *Search programs and files*. If the platform does not support eVM, you will see that VT-x is not supported. In Figure D. 1, the hardware platform is found to support neither VT-x nor VT-d.

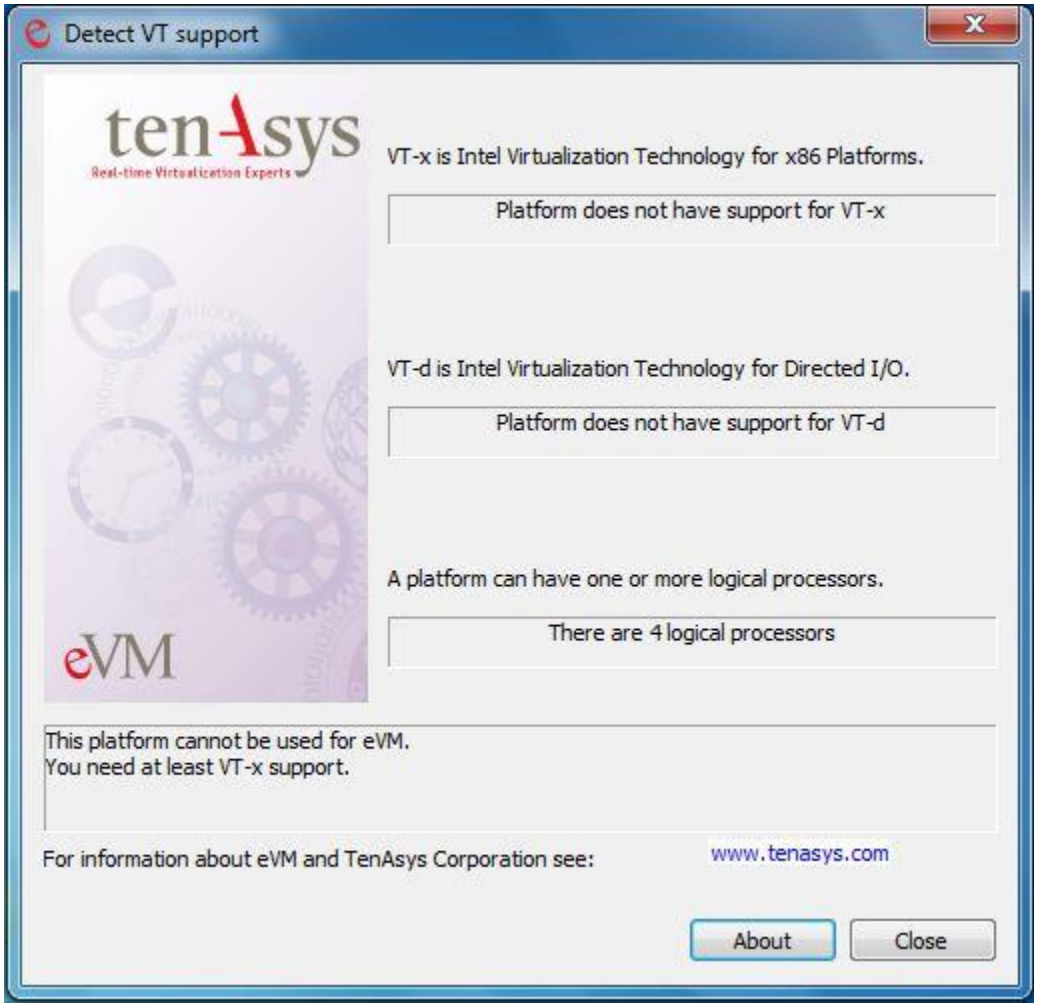

Detect VT support
tenAsys
Real-time Virtualization Experts
eVM
VT-x is Intel Virtualization Technology for x86 Platforms.
Platform does not have support for VT-x
VT-d is Intel Virtualization Technology for Directed I/O.
Platform does not have support for VT-d
A platform can have one or more logical processors.
There are 4 logical processors
This platform cannot be used for eVM.
You need at least VT-x support.
For information about eVM and TenAsys Corporation see:
www.tenasys.com
About
Close

Figure D. 1 - VtProbe, Hardware Platform Doesn't Support eVM

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

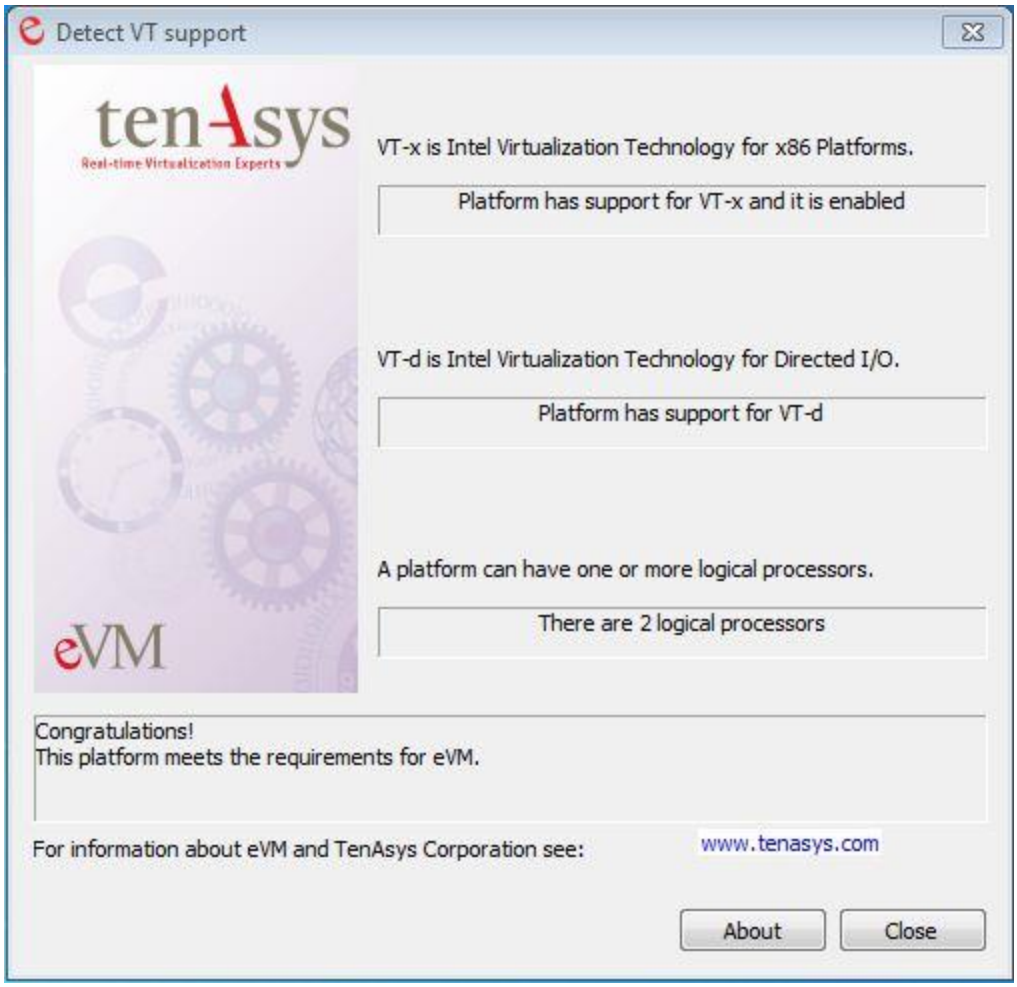

Detect VT support
tenAsys
Real-time Virtualization Experts
eVM
VT-x is Intel Virtualization Technology for x86 Platforms.
Platform has support for VT-x and it is enabled
VT-d is Intel Virtualization Technology for Directed I/O.
Platform has support for VT-d
A platform can have one or more logical processors.
There are 2 logical processors
Congratulations!
This platform meets the requirements for eVM.
For information about eVM and TenAsys Corporation see:
www.tenasys.com
About
Close

Figure D. 2 - VtProbe: Hardware Supports Virtualization Technology

In Figure D. 2, the hardware platform was found to support both VT-x and VT-d.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

E. Modifying device drivers for bus-master devices

Any device which generates bus cycles to transfer data is known as a bus master. Such a device is programmed with physical memory addresses at which it can read or write data. When a device driver running in the guest operating system programs such a device, either the platform must translate the physical addresses to direct the read/write operation to the correct physical memory address or the device driver must be modified to make this translation explicitly. This appendix describes how to make this modification.

E.1 Discovering the guest physical base address

The guest physical base address is the offset at which the guest is located on the host. This offset must be added to physical addresses programmed into any bus master device. This offset is discovered using an interface to the VMM via the VMCALL instruction. The VMCALL instruction causes an exit to the VMM which then processes the request based on the contents of the CPU registers. To find the guest physical address, execute the VMCALL instruction with EAX set to the value 0x80020001. If the call succeeds, then the carry flag will be clear, and the guest physical base address is returned in register EBX. The size of the memory allocated to the guest will be returned in register ECX. The VMCALL instruction is described in detail in Intel documentation.

E.2 Using the guest physical base offset value

Once the base address has been obtained, it can be used to offset any physical address generated by the device. Anywhere in a device driver where an address is provided which the device will use to generate a physical address has to be modified in this way.

A pseudocode implementation of a function to implement this might look like this (OsGetPhysicalAddress is assumed to be the operating system's regular call):

```
Paddr_t LibGetRealPhysicalAddress(PVOID *logical_address)
{
   Paddr_t real_address = OsGetPhysicalAddress(logical_address);
   Uint32 sys_offset ;
   Uint32 regs[4];

   regs[0] = GETMEMPARAMS;
   if (do_vmcall(regs) == 0)
       sys_offset = regs[1];
   else
       sys_offset = 0;

   return real_address + sys_offset;
}
```

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

F. eVM API calls

The eVM API is a Windows library which allows the user to control or configure certain features from within a Windows application. Included in the API is also the means of accessing the shared memory area between Windows and the eVM Guest software.

F.1 Include the API in your application

To use the API, add the following settings to your Visual Studio C/C++ project:
Compiler settings: Additional include directories: "$(EVM)\nt\include"
Linker settings: General: Additional library directories: "$(EVM)\nt\lib"
Linker settings: Input: Additional dependencies: evmapi.dll (or evmapi64.dll for 64-bit applications)

In your source code, add **#include <evmapi.h>**

F.2 Types

`EVMSTATUS`

The eVM API status type.

`EVMLOCATION`

The eVM API location type. Identifies an eVM node.

F.3 API Calls

General calls

evmGetLastError

EVMSTATUS evmGetLastError();

Returns the status of the last failed API call. If an eVM API call fails, use this call to return the failure status code.

Command and Control calls

The following APIs are proposed for control of the eVM VMM, and its configured guests.

evmStartVirtualMachine

EVMSTATUS evmStartVirtualMachine(NTXLOCATION evmLoc);

Starts the VMM on a given node, where vmmLoc is the location of the node (returned from **evmGetFirstVmm**/**evmGetNextVmm**). This call loads and starts the VMM but does not start a guest, unless a default guest is configured.

evmStopVirtualMachine

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

EVMSTATUS evmStopVirtualMachine(EVMLOCATION evmLoc);

Stops the VMM on a given node, where *evmLoc* is the location of the node. This call first stops a guest running on the VMM then stops the VMM.

evmStartGuest

EVMSTATUS evmStartGuest(EVMLOCATION evmLoc, LPTSTR sGuestName);

Starts a guest on the VMM indicated by the vmmLoc parameter, and the guest configuration name. The name corresponds to a configuration generated by the configuration utility.

evmStopGuest

EVMSTATUS evmStopGuest(EVMLOCATION evmLoc);

Stops a guest on the VMM indicated by the vmmLoc parameter.

evmGetVMMStatus

EVMSTATUS evmGetVMMStatus(LOCATION vmmLoc, LPTSTR Buffer, DWORD BufferLength);

Returns the running state of the VMM indicated by the *vmmLoc* parameter, and returns the name of the running configuration, if any.

VMM Management calls

These calls are implemented using FindFirstFile, FindNextFile with filter *.cfg in the %EVMCFG% directory.

evmGetFirstVmm

LOCATION evmGetFirstVmm(void);

Returns a location value for the first VMM instance.

evmGetNextVmm

LOCATION evmGetNextVmm(LOCATION evmLoc);

Returns a location value for the next VMM instance found after the one specified by the parameter.

Configuration Management calls

evmGetFirstGuest

EVMHANDLE evmGetFirstGuest(LPTSTR Buffer, DWORD BufferLength);

Returns a handle and the name of the first guest configuration found.

evmGetNextGuest

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

EVMHANDLE evmGetNextGuest(EVMHANDLE LastGuest, LPTSTR Buffer, DWORD BufferLength);

Returns a handle and the name of the next guest configuration after the last one found.

evmGetDefaultGuest

BOOL evmGetDefaultGuest(LPTSTR Buffer, DWORD BufferLength);

Returns the name of the default guest configuration. Bufferlength must be set before the call to the size of the available buffer. If the guest name returned is too long for the buffer then the call will fail and a suitable status returned.

evmSetDefaultGuest

```
EVMHANDLE evmSetDefaultGuest(LPTSTR Buffer, DWORD BufferLength);
```

Sets the default guest name. If Buffer is NULL then the default guest name is removed, so there is no default guest. If a default guest already exists then it is overwritten.

Shared Memory Management

In v1.0 there will be a single shared-memory buffer defined per node for use as a communications buffer between a Windows application and the guest. The following calls are used to manipulate the shared memory from Windows, using the standard NTX memory mapping calls to map a segment catalogued in the VMM process.

evmGetSharedMemoryInfo

EVMSTATUS evmGetSharedMemoryInfo(EVLOCATION evmLoc);

Returns basic information about the shared memory buffer. Includes at least the size of the buffer.

evmMapSharedMemory

PVOID evmMapSharedMemory(EVMLOCATION evmLoc);

This call maps all of the configured shared memory to the calling process.

evmUnmapSharedMemory

EVMSTATUS evmUnmapSharedMemory(EVMLOCATION evmLoc);

Unmaps shared memory from an application where it was previously mapped.

evmSignalSharedMemory

EVMSTATUS evmSignalSharedMemory(EVMLOCATION evmLoc);

Signal an internal semaphore associated with the shared memory area. This results in a signal being sent to the guest software on the node indicated by *evmLoc*.

Copyright © 2014 TenAsys Corporation,
All Rights Reserved

evmWaitForSharedMemory

EVMSTATUS evmWaitForSharedMemory(EVMLOCATION evmLoc, DWORD dwMilliseconds);

Wait for a signal from the semaphore associated with the shared memory buffer, or time out after dwMilliseconds.